



Generalized Equivalence Checking of Concurrent Programs

Benjamin Bisping



<https://bbisping.de>



This document is a preprint of a thesis to be submitted for the degree of Doctor of Natural Sciences (Dr. rer. nat.) at the Department of Electrical Engineering and Computer Science, Technical University of Berlin. The final version will be published on the TU Berlin publication server with open licensing.

The most current version of this document is available on
<https://generalized-equivalence-checking.equiv.io/>.

The source code of this document is available on
<https://github.com/benkeks/generalized-equivalence-checking>.

Feel free to send suggestions to info@bbisping.de!

This PDF was generated on 2025-07-19.

Summary

We study the *spectroscopy problem* that asks which notions from a spectrum of behavioral equivalences relate a pair of states in a transition system. This allows a generalized handling of questions on *how equivalent two programs are*.

As *main result*, we solve the spectroscopy problem for *finite-state systems* and a hierarchy of semantic models known as the *weak linear-time–branching-time spectrum*, due to van Glabbeek. The spectrum arises because of different ways of understanding nondeterminism and internal behavior in models of concurrent programs. We also treat the *strong spectrum* (without internal behavior) as well as use cases of generalized equivalence checking in verification and concurrency theory.

Our *approach* relies on a *quantitative understanding of spectra* in terms of how many syntactic features of Hennessy–Milner modal logic are used to characterize equivalences.

As *key trick* to solve the spectroscopy problem, we prove spectra of equivalence to be captured by *spectroscopy games* where energy budgets bound the features an attacker may use to express differences of states. Optimal attack strategies correspond to *distinguishing formulas* with a minimal usage of syntax. The spectroscopy problem thus reduces to the problem of computing *minimal attacker-winning budgets* in spectroscopy games. For this, we provide an algorithm to compute such budgets on a wider class of *Galois energy games*. The resulting spectroscopy algorithms are exponential for the PSPACE-hard spectrum of equivalence problems, but can be instantiated to polynomial-time solutions on the P-easy part.

Aiming for *applicability*, we implement the spectroscopy procedure in the web tool equiv.io, which continues a tradition of *concurrency workbenches*. Using it, we can *check for dozens of equivalences* at once. We apply the spectroscopy approach to small *case studies from verification and translation of concurrent programs*. Core parts of the thesis are supported by an *Isabelle/HOL formalization*.

Zusammenfassung

Wir befassen uns mit dem *Spektroskopie-Problem*. Dieses fragt, welche Verhaltensgleichheiten eines Spektrums Zustände in einem Transitionssystem in Beziehung setzen. Damit wird es möglich, die Frage, *wie äquivalent zwei Programme sind*, allgemein zu handhaben.

Als *Hauptergebnis* lösen wir das Spektroskopie-Problem für Systeme mit endlichem Zustandsraum und eine Hierarchie semantischer Modelle, die als „Schwach-Linearzeit-Verzweigungszeit-Spektrum“ nach van Glabbeek bekannt ist. Dieses Spektrum ergibt sich aus verschiedenen Wegen, Nichtdeterminismus und internes Verhalten in Modellen nebenläufiger Programme zu fassen. Wir behandeln auch das „Starke Spektrum“ (ohne internes Verhalten) sowie diverse Anwendungsfälle generalisierter Äquivalenzprüfung in Verifikation und Nebenläufigkeitstheorie.

Unser *Ansatz* ruht auf einem *quantitativen Verständnis von Spektren* darin, wie viele syntaktische Möglichkeiten in Hennessy-Milner-Modallogik zur Charakterisierung verwendet werden.

Der *zentrale Trick* zur Lösung des Spektroskopie-Problems ist, zu zeigen, wie Gleichheitsspektren durch *Spektroskopie-Spiele* abgedeckt werden. In diesen begrenzen Energiebudgets die Möglichkeiten eines Angreifers, der Unterschiede zwischen Zuständen ausdrücken möchte. Optimale Angriffsstrategien korrespondieren zu *unterscheidenden Formeln*, die minimale syntaktische Möglichkeiten nutzen. Das Spektroskopie-Problem reduziert sich damit auf die Berechnung *minimaler Budgets*, mit denen der Angreifer in Spektroskopie-Spielen gewinnt. Hierzu liefern wir einen Algorithmus, mit dem sich solche Budgets für eine größere Klasse von *Galois-Energie-Spielen* ermitteln lassen. Die sich ergebenden Spektroskopie-Algorithmen sind exponentiell für das PSPACE-harte Spektrum von Gleichheitsproblemen, aber lassen sich zu polynomiellen Lösungen für den P-einfachen Teil instanziiieren.

Mit dem Ziel der *Anwendbarkeit* implementieren wir die Spektroskopie im Web-Tool equiv.io. Dieses Programm folgt einer Tradition von *Concurrency-Workbenches*. Mit ihm lassen sich *dutzende Gleichheiten auf einmal prüfen*. Wir wenden den Spektroskopie-Ansatz in kleinen Fallbeispielen aus Verifikation und Übersetzung nebenläufiger Programme an. Zentrale Teile der Arbeit werden durch eine *Isabelle/HOL-Formalisierung* gestützt.

To the memory of **Mark Alexander Sibly**,
creator of the programming language in which I wrote my first
concurrent program, and a profound influence on me
and so many others!

Acknowledgments

This research project started due to Covid, in a way. Because of the pandemic, CONCUR’20 was taking place as an online event. And one afternoon, after listening to the talks, sitting alone in my office, I started to [implement a small idea](#) on generalizing the bisimulation game. Last year, the office was closed—together with the whole building—because of rusty pipes. Unfortunately, that’s not the only thing that’s been going wrong, since. Crazy times!

That this research project has still reached the stage of a PhD thesis, is thanks to many great people (sorted by category, mostly in order of appearance).

- **Environment:** Uwe Nestmann and my colleagues at the Modelle und Theorie Verteilter Systeme group, especially my lovely office mates Edgar Arndt and Valeria Zahoransky. Alex Wiemhoefer, who offered a desk in her office when mine was closed. My “Sunday friends” and my parents. You’re all amazing!
- **Hospitality:** The organizers of the VTSA Summer School 2021 in Liège. Rob van Glabbeek and the Laboratory for Foundations of Computer Science for my 2023 stay in Edinburgh. Julia (and Fréd and kids) as well as Rosi for hosting me during visits in Paris and Oldenburg. David N. Jansen together with Lijun Zhang’s group at the Institute of Software, Chinese Academy of Sciences for my 2024 stay in Beijing.
- **Collaboration:** David N. Jansen, who discovered and corrected crucial flaws in my ideas. Caroline Lemke, who found out why “my” energy games are indeed decidable. And many more marvelous students and collaborators (listed in Section 1.3).
- **Co-writing:** Schrippe, Baum, Alina, and Anna Sophia.
- **Feedback** on parts of this thesis: Nadine Karsten, Caroline Lemke, Andrei Aleksandrov, Dominik Geißler, Uli Fahrenberg, Karl Mattes, Jeroen Keiren, Uwe Nestmann, Amelie Heindl, Gabriel Vogel, Sebastian Wolf, David Karcher, David N. Jansen.

Table of contents

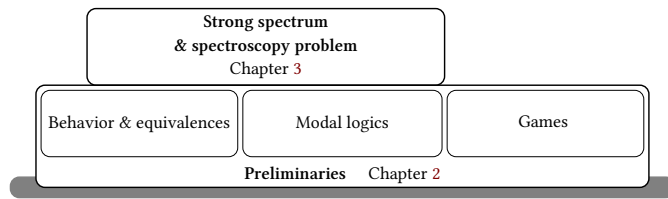
I	Our Problem	1
1	Introduction: What's the Difference?	3
1.1	Linear-Time–Branching-Time Spectroscopy	3
1.2	This Thesis	7
1.3	Artifacts and Papers	9
2	Preliminaries: Communicating Systems and Games	13
2.1	Behavior of Programs	14
2.1.1	Labeled Transition Systems	14
2.1.2	Calculus of Communicating Systems	16
2.2	Behavioral Equivalences	18
2.2.1	Trace Equivalence	19
2.2.2	Similarity and Bisimilarity	20
2.2.3	Equivalence Hierarchies	21
2.2.4	Congruences	22
2.2.5	Quotient Systems and Minimizations	24
2.3	Modal Logics	25
2.3.1	Hennessy–Milner Logic to Express Observations . . .	25
2.3.2	Characterizing Bisimilarity via HML	26
2.3.3	The Perks of Modal Characterizations	27
2.3.4	Expressiveness and Distinctiveness	28
2.4	Games	29
2.4.1	Reachability Games	29
2.4.2	The Semantic Game of HML	31
2.4.3	The Bisimulation Game	32
2.4.4	Deciding Reachability Games	35
2.4.5	How Bisimulation Game and HML Are Connected . .	37
2.5	Discussion	40
3	Context: The Spectrum of Equivalences	43
3.1	Observability Hierarchies	43
3.1.1	Understanding the Equivalence Hierarchy through Modal Logics	44
3.1.2	Incomparabilities	45
3.1.3	Lattices	45
3.2	The Linear-Time–Branching-Time Spectrum	47
3.2.1	Spectra as Observability Lattices	47
3.2.2	Strong Notions of Observability	48
3.2.3	The Strong Linear-Time–Branching-Time Spectrum .	51
3.2.4	Any Questions?	55

3.2.5	Non-Intersectionality	56
3.3	Spectroscopy	57
3.3.1	The Spectroscopy Problem	57
3.3.2	Spectroscopy as Abstract Subtraction	59
3.3.3	Complexities	60
3.4	Discussion	61
II	Generalized Equivalence Checking	63
4	Approach: Equivalence Problems as Energy Games	65
4.1	Energy Games	66
4.1.1	Monotonic Energy Games	66
4.1.2	Declining Energy Games	70
4.2	Characterizing the P-easy Part of the Spectrum	71
4.2.1	The P-easy Slice	71
4.2.2	The Bisimulation <i>Energy</i> Game	72
4.2.3	Correctness of Characterization	74
4.3	Deciding Energy Games	77
4.3.1	Galois Connections	77
4.3.2	The Algorithm	79
4.3.3	Complexity	82
4.3.4	Solving Declining Energy Games	82
4.3.5	Polynomial Spectroscopy Complexity	86
4.4	Discussion	86
5	Spectroscopy of the Strong Equivalence Spectrum	89
5.1	The Strong Spectroscopy Game	90
5.1.1	The Game	90
5.1.2	Correctness	94
5.1.3	Complexity	95
5.2	Clever Games on Subgraphs	96
5.2.1	Pruning with Logic	96
5.2.2	The Clever Game	98
5.3	Deciding Individual Equivalences	99
5.3.1	Deriving Equivalence Games	100
5.3.2	Regaining Polynomiality in Derived Games	102
5.4	Discussion	104
III	... of Concurrent Programs	107
6	Recharting the Weak Silent-Step Spectrum	109
6.1	Weak Equivalences in General	110
6.1.1	Silent Transitions	110
6.1.2	Weak Traces and Weak Bisimulation	111
6.1.3	HML of Stability-Respecting Branching Bisimilarity	113
6.2	Case Studies—and the Need for Other Weak Equivalences	114
6.2.1	Parallelizing Compilers—and Contrasimulation	114
6.2.2	Abstraction as Congruence—and Stable Failures	117

6.3	Expressing the Weak Spectrum by Quantities	118
6.3.1	Syntactic Expressiveness	118
6.3.2	Weak Spectrum	119
6.4	Discussion	126
7	Spectroscopy for the Weak Spectrum	127
7.1	The Weak Spectroscopy Game	127
7.1.1	The Game	127
7.1.2	Correctness	130
7.1.3	Isabelle/HOL Formalization	133
7.1.4	Complexity	137
7.2	Tackling Our Case Studies	138
7.2.1	Parallelizing Compilers	138
7.2.2	τ -Abstraction and Failures	139
7.3	Variants	140
7.3.1	Optimizing Branching Conjunctions	140
7.3.2	Covering Revivals and Decorated Traces	142
7.3.3	Extending to Other Equivalences	143
7.4	Discussion	144
IV	... and Beyond	147
8	Implementations	149
8.1	Prototype: <i>equiv.io</i>	149
8.1.1	Usage	150
8.1.2	Application to Peterson's Mutual Exclusion Protocol	152
8.1.3	Program Structure	157
8.1.4	Benchmarks	159
8.2	Student Implementations	161
8.2.1	Computer Game: <i>The Spectroscopy Invaders</i>	162
8.2.2	CAAL Extension	164
8.2.3	GPU Implementation: <i>gpuequiv</i>	164
8.3	Discussion	166
9	Conclusion	171
	References	175
	Notation	187

Part I

Our Problem



1 Introduction: What's the Difference?

Have you ever looked at two program descriptions and wondered *how equivalent they are*—or, conversely, how they can be distinguished?

In concurrency theory, one runs into this problem often, for instance, when analyzing models of distributed algorithms or when devising examples for teaching. More mundanely, the question occurs every time that someone rewrites a program part and hopes for it to still do its job.

My first formal encounter with the question of behavioral equivalence came when I was implementing a translation from the process algebra Timed CSP to Timed Automata.¹ How to tell whether the translation would properly honor the semantics of the two formalisms? Did it translate CSP terms to automata *with the same meaning*? Even the definition of the question is tricky, as there are different notions of what counts as “same meaning” in the semantics of programs.

I then took my very first look into seminal work on the landscape of process equivalences: the “linear-time-branching-time spectrum” by van Glabbeek (1990, 1993; 2001 with a combined 2500 citations on Google Scholar). A central figure, reproduced in Figure 1.1, mesmerized me. So many equivalences! But, how to find out for a given pair of processes which of the many equivalences apply? Over the years, I have learned that others, too, have run into the not-quite-straightforward question which equivalences hold: For instance, Nestmann & Pierce (2000) thinking about process algebra encodings and Bell (2013) verifying compiler optimizations. We share an itch.

Our problem can abstractly be summarized as follows:

How does one conveniently decide for a pair of systems which notions from a spectrum of behavioral equivalences equate the two?

The above question will be the *research question of this thesis*. We want to enable future researchers to tap into the wisdom of the linear-time–branching-time spectrum and to easily determine what equivalences fit their models.

1.1 Linear-Time-Branching-Time Spectroscopy

To illustrate the problem, let us look at an example that we will be able to solve via tool support at the end of the thesis in Section 8.1.2. It includes numerous concepts we will define and discuss more deeply on our way there.

¹ My job as student research assistant was to bridge between the tools FDR2 and UPPAAL for Göthel (2012). The previous work I was to base the translations on had serious flaws: One approach introduced spurious deadlocks to the model, the other was unable to handle nesting of choices and parallel composition. Clearly, we had to change the encoding!

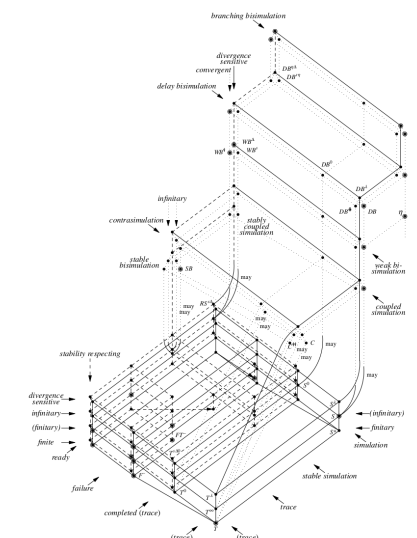


Figure 1.1: The linear-time-branching-time spectrum with silent moves as depicted in van Glabbeek (1993). (Do not try to read this small figure! Its role here is that of a symbol. This thesis will use a simpler version in Figure 6.5.)

Example 1.1 (Verifying Peterson's mutual exclusion). Many verification tasks can be understood along the lines of “how equivalent” two models are. The models can usually be expressed as labeled transition systems, that is, as graphs where nodes represent program *states* and edges represent *transitions* between them, labeled by *actions*.

Figure 1.2 replicates a standard example, known, for instance, from the textbook *Reactive systems* (Aceto et al., 2007): The left-hand side gives a graph specification of mutual exclusion Mx as two users A and B entering their critical section ec_A/ec_B and leaving lc_A/lc_B before the other may enter. The right-hand side shows the transition system of Peterson's mutual exclusion algorithm (1981) starting in Pe, with internal steps \rightarrow due to the coordination that needs to happen.² For Pe to faithfully implement mutual exclusion, it should behave somewhat similarly to Mx.

² To fit on the page, Pe is minimized by a behavioral equivalence that will be important later on: stability-respecting branching bisimilarity.

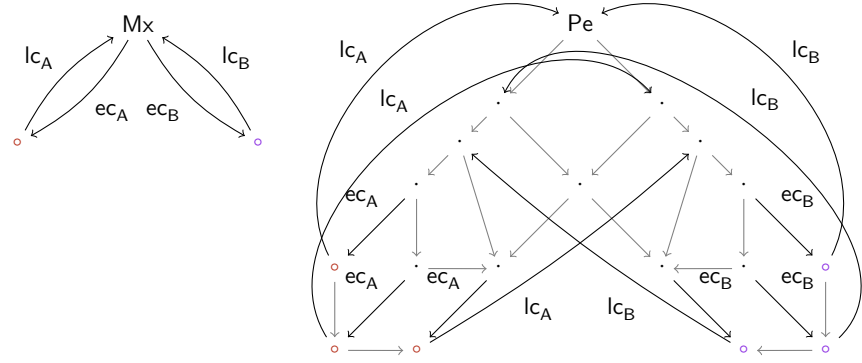


Figure 1.2: A specification of mutual exclusion Mx, and Peterson's protocol Pe, minimized with respect to stability-respecting branching bisimilarity.

In Section 8.1.2, we will see how the transitions of Pe derive from pseudocode and formal model.

Semantics in concurrent models must treat nondeterminism and internal steps in some way. As we will see throughout this thesis, setting the degree to which nondeterminism counts induces equivalence notions with subtle differences. In the example, Pe and Mx *weakly simulate* each other, intuitively meaning that a tree of options passing over internal activity from one process can be matched by a similar tree of the other. This implies that they have the same *weak traces*, that is, matching paths. However, they are not weakly *bi-similar*, which would require a higher degree of symmetry than mutual similarity, namely, matching absence of options. There are many more such notions, which can be incomparable in how they relate processes. In our example, one might wonder: Are there notions relating Pe and Mx *besides* mutual weak similarity?

State of the art. For many of the existing behavioral preorders and equivalences, there are various algorithms and implementations to decide whether processes are equivalent (see [Garavel & Lang, 2022](#) for a survey). So, it would be an option to throw an array of algorithms on the transition systems of Example 1.1. For example, let us pick out stable failures and contrasimilarity, which are close to weak similarity in Figure 1.1. CAAL ([Andersen, Andersen, et al., 2015](#)) could establish mutual weak similarity, but does not support the other notions. mCRL2 ([Bunte et al., 2019](#); [Groote & Mousavi, 2014](#)) would rule out stable-failure equivalence. But for contrasimilarity, we would need to implement our own solution because there is no tool supporting it. Combining different tools and algorithms like this is tiresome and prone to subtle errors. It would be desirable to treat the question in *just one* algorithm!

Our offer. This thesis describes how to *decide all equivalences* in one *uniform approach* based on *energy games*, which is *easily implemented in tools*.

As we will discuss in more detail in Section 8.1.2, our accompanying tool [equiv.io](#) answers the question which equivalences apply for Example 1.1 in a small query. Here is an excerpt:

```
Pe = (A1 | B1 | TurnA | ReadyAf | ReadyBf)
    \ {readyAf, readyAt, setReadyAf, setReadyAt, readyBf, readyBt,
       setReadyBf, setReadyBt, turnA, turnB, setTurnA, setTurnB}
Mx = ecA.lcA.Mx + ecB.lcB.Mx

@compareSilent Pe, Mx
```

[Interactive model on equiv.io.](#)

This query leads to the output in Figure 1.3,³ which is to be understood as an interactive version of the linear-time–branching-time spectrum of Figure 1.1. The big blue circle ● in Figure 1.3 indicates that weak simulation indeed is the most specific equivalence to equate Pe and Mx. Lines mean that the notion at the top implies the one below. Blue half-circles ◐ indicate preordering in only one direction. For instance, Pe is *preordered* to Mx with respect to two more specific notions, namely *η-simulation* and *stable simulation* (and notions below). Intuitively, this means that Pe *implements* Mx more faithfully than pure weak similarity would demand. Red triangles ◀ mark distinctions.

By running *one* simple algorithm, the tool decides *twenty-six* equivalence (and preorder) problems on Pe and Mx (in about 150 ms on my laptop).

Spectroscopy analogy. How does the tool decide so many equivalences in one shot? The key are van Glabbeek’s “linear-time–branching-time spectrum” papers on comparative concurrency semantics ([1990, 1993](#)). They treat a zoo of distinct *qualitative questions* of the form “Are processes *p* and *q* equivalent with respect to notion *N*?”, where *N* would, for example, be trace or bisimulation equivalence. The papers unveil an underlying structure where equivalences can easily be compared with respect to their distinctive power. This is analogous to the *spectrum of light* where seemingly qualitative properties

³ This thesis often provides snippets for models on [equiv.io](#). You can examine and modify the example models by clicking on the links reading [Interactive model on equiv.io](#).

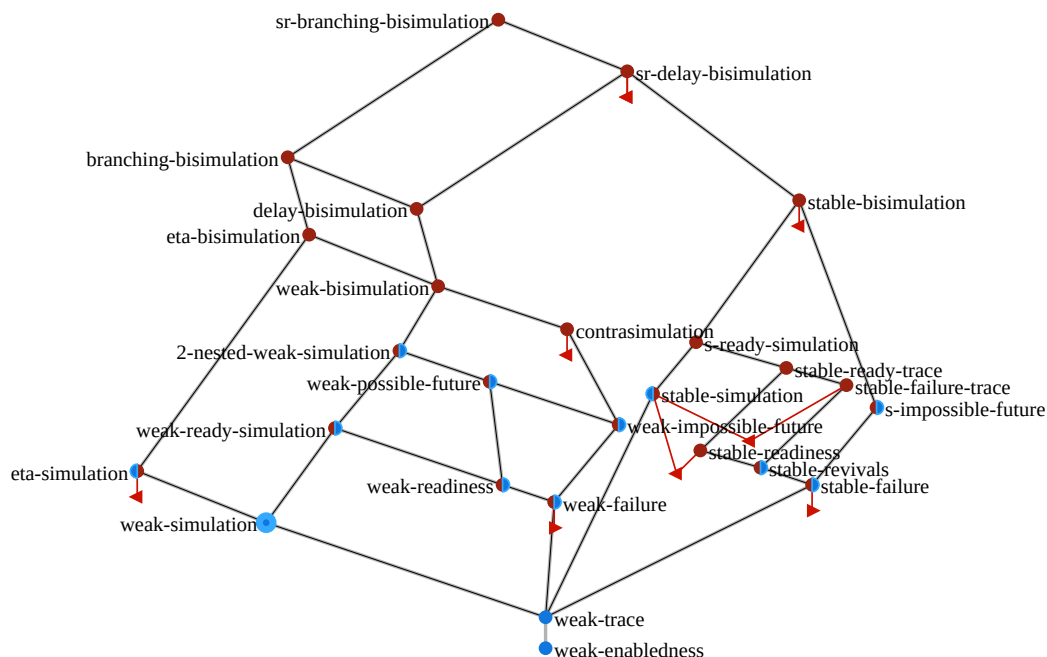


Figure 1.3: Output of equiv.io about weak equivalences to relate Pe to Mx of Example 1.1. Where notions are connected, the one at the top implies the one underneath.

(“The light is blue / green / red.”) happen to be *quantitative* (“The distribution of wavelengths peaks at 460/550/630 nm.”).

For light (i.e. electromagnetic radiation), the mix of wavelengths can be determined through a process called *spectroscopy*. So, we could reframe the question behind this thesis also:

i Idea 1: A Spectroscopy for the Spectra

If there are “linear-time-branching time spectra,” does this mean that there also is some kind of “linear-time-branching-time *spectroscopy*”?

The difference that this thesis makes. We answer the spectroscopy question positively, which is the key step to tackle our research question: One can compute what mix of (in-)distinguishabilities exists between a pair of finite-state processes, and this places the two on a spectrum of equivalences. We thus turn a set of *qualitative* equivalence problems into one *quantitative* problem of *how equivalent* two systems are. This amounts to an abstract form of subtraction between programs, determining what kinds of differences an outside examiner might observe. Thereby, *one* algorithm works for *all* of a spectrum.

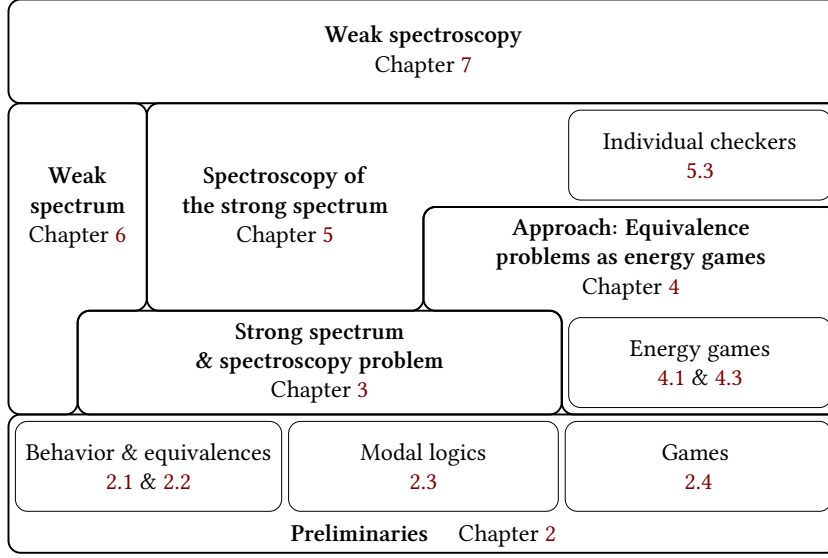


Figure 1.4: The *theory stack* of this thesis. Chapters and ideas build upon those underneath.

1.2 This Thesis

At the core, this thesis presents an algorithm to *decide all behavioral equivalences at once* for varying spectra of equivalence using *energy games to limit possible distinctions* through attacker budgets. More precisely, we make the following contributions, each coming with a “side quest:”

- Chapter 2 lays some foundations and makes precise *how bisimulation games relate to grammars of distinguishing formulas* from Hennessy–Milner modal logic.
 - Main result: Attacker strategies in the bisimulation game \mathcal{G}_B correspond to distinguishing formulas in a subset $\mathcal{O}_{[B]}$ of modal logic HML (Theorem 2.3).
 - Side quest: Certifying algorithms to check equivalences.
- Chapter 3 shows *how to understand the strong linear-time–branching-time spectrum quantitatively* and formalizes the *spectroscopy problem*.
 - Main result: The spectroscopy problem (Problem 1) is PSPACE-hard for the strong spectrum N^{strong} .
 - Side quest: Equivalence checking as subtraction.
- Chapter 4 introduces the approach of *characterizing equivalence spectra through energy games* and *how to decide such games*.
 - Main result: The preorder and equivalence problems in the P-easy part of the strong spectrum N^{peasy} are characterized by an energy game version of the bisimulation game \mathcal{G}_B^\star (Theorem 4.1).

- Side quest: Deciding Galois energy games (Problem 2).
- Chapter 5 applies the approach to decide the whole strong spectrum through one *game for linear-time-branching-time spectroscopy*.
 - Main result: The strong spectroscopy game \mathcal{G}_Δ covers the strong spectrum $\mathbf{N}^{\text{strong}}$ (Theorem 5.1) in exponential time.
 - Side quest: Deriving efficient *individual equivalence checkers*.
- Chapter 6 recharts the *weak spectrum of equivalences accounting for silent steps* to fit our game approach.
 - Main result: The Hennessy–Milner logic subset HML_{SRBB} characterizes stability-respecting branching bisimilarity (Theorem 6.1), and its sublogics describe the weak spectrum \mathbf{N}^{weak} .
 - Side quest: Case studies in concurrency theory research.
- Chapter 7 adapts the game for the *weak spectrum of equivalences*.
 - Main result: The weak spectroscopy game \mathcal{G}_∇ characterizes the weak spectrum \mathbf{N}^{weak} (Theorem 7.1).
 - Side quest: Isabelle/HOL formalization.
- Chapter 8 showcases four *implementations* to conveniently perform equivalence spectroscopies in web browsers and reports some benchmarking results.
 - Main result: Our approach is more general than what is usually found in tools and can compete with algorithms for simulation-like notions.
 - Side quest: Analyzing Peterson's mutual exclusion protocol.

Each chapter ends with a discussion of its position in the context of the thesis and of related work. Chapter 9 recapitulates how the thesis ascends through a *hierarchy of game characterizations* and considers the wider picture. Recurring symbols can be looked up under **Notation** on page 187.

Figure 1.4 gives an overview of how parts of this thesis build upon each other. An example of how to read the figure: Chapter 4 describes the approach to frame equivalence problems as energy games. It builds on a part of the modal characterization for the strong spectrum in Chapter 3 and on game theory from Chapter 2, which is adapted through energy games.

Not this thesis. We limit ourselves to the relevant parts of the *strong spectrum* (van Glabbeek, 1990) and the *weak spectrum* (van Glabbeek, 1993). For instance, this excludes questions around value-passing, open/late/early bisimilarities, and barbed congruences on the π -calculus (cf. Sangiorgi, 1996). Also, we do not consider timed or probabilistic equivalences (cf. Baier et al., 2020), nor behavioral distances (Fahrenberg & Legay, 2014). Neither do we aim to re-survey behavioral equivalences in encyclopedic detail, which means that several notions will be covered without a detailed discussion of their features and merits.

1.3 Artifacts and Papers

This thesis ties together the work of several publications. It is written to be understandable on its own. For details, we typically refer to the original publications or to other artifacts for implementation and machine-checked proofs.

Publications. The following four publications (with me as main author) fuel the following chapters:

- “Deciding all behavioral equivalences at once: A game for linear-time–branching-time spectroscopy” (LMCS 2022, with Nestmann & Jansen) introduces the *spectroscopy problem* and the *core idea* to decide the whole strong spectrum using games that search trees of possible distinguishing formulas. (Conference version: TACAS 2021)
- “Process equivalence problems as energy games” (CAV 2023b) makes a *big technical leap by using energy games*, which removes the necessity for explicit formula construction. (Tech report: arXiv 2023c)
- “Characterizing contrasimilarity through games, modal logic, and complexity” (Information & Computation 2024, with Montanari) closes *gaps in the weak spectrum of equivalences* for complexity, games, and their links to modal logics. (Isabelle/HOL theory: AFP 2023; Workshop version: EXPRESS/SOS 2021)
- “One energy game for the spectrum between branching bisimilarity and weak trace semantics” (EXPRESS/SOS 2024, with Jansen) adapts the *spectroscopy approach for the weak spectrum*. (Journal version: Preprint 2025)

Prototype. The algorithms of this thesis have been validated through a *Scala prototype implementation*. Throughout the thesis, many examples come with listings of equiv.io input and a link to try it out in the browser. You have already seen one such example for Peterson’s mutex in Section 1.1. Section 8.1.1 explains the usage of equiv.io. The source is available openly on <https://github.com/benkeks/equivalence-fiddle>.

Isabelle formalization. In order to relieve the document of technical proofs, these are contained in machine-checkable *Isabelle/HOL theories*.

- “A weak spectroscopy game to characterize behavioral equivalences” formalizes the core results of Chapter 6 and Chapter 7: <https://equivio.github.io/silent-step-spectroscopy>. The theory (Barthel et al., 2025) has been developed together with TU Berlin students Barthel, Hübner, Lemke, Mattes, and Mollenkopf. Also, many theorems of earlier chapters can be understood to be supported by the formalization.
- “Equivalence spectrum formalization” supplies proofs for Chapter 2 and Chapter 3: <https://benkeks.github.io/lbt-spectroscopy-isabelle/>.

Student theses. Some parts of this thesis strongly rely on student work that I have supervised, in particular, on the following theses:

- Trzeciakiewicz (2021): “*Linear-time–branching-time spectroscopy as an educational web browser game*” provides a computer game version of the spectroscopy procedure to be discussed in Section 8.2.1.
- Ozegowski (2023): “*Integration eines generischen Äquivalenzprüfers in CAAL*” extends the Concurrency Workbench Aalborg Edition with a spectroscopy feature, reported in Section 8.2.2, extended by Straßnick & Ozegowski (2024).
- Mattes (2024): “*Measuring expressive power of HML formulas in Isabelle/HOL*” proves the approach to modal characterization of Section 3.2.
- Vogel (2024): “*Accelerating process equivalence energy games using Web-GPU*,” topic of Section 8.2.3, allows massive parallelization of key parts of our algorithm on the GPU.
- Lemke (2024): “*A formal proof of decidability of multi-weighted declining energy games*” formalizes the *Galois energy games* of Section 4.3. Its Isabelle theory is archived on <https://github.com/crmrtz/galois-energy-games>.

Other publications. My prior publications are only indirectly connected to this PhD thesis. Still, they lead me to topics and techniques:

- Bisping, Brodmann, Jungnickel, Rickmann, Seidler, Stüber, Wilhelm-Weidner, Peters, Nestmann (ITP 2016): “Mechanical verification of a constructive proof for FLP.”
- Bisping & Nestmann (TACAS 2019): “Computing Coupled Similarity.”
- Bisping, Nestmann, Peters (Acta Informatica 2020): “Coupled similarity: The first 32 years.”

Other student theses. During this research project, I have also supervised several other Bachelor theses, many of which played important roles in shaping the research. Although most of them do not appear directly on the following pages, I want to acknowledge the students’ vital contributions to this research.

- Peacock (2020): “*Process equivalences as a video game.*”
- Lê (2020): “*Implementing coupled similarity as an automated checker for mCRL2.*”
- Wrusch (2020): “*Ein Computerspiel zum Erlernen von Verhaltensäquivalenzen.*”
- Reichert (2020): “*Visualising and model checking counterfactuals.*”
- Wittig (2020): “*Charting the jungle of process calculi encodings.*”
- Bulik (2021): “*Statically analysing inter-process communication in Elixir programs.*”

- Montanari (2021): “Kontrasimulation als Spiel.”
- Pohlmann (2021): “Reducing strong reactive bisimilarity to strong bisimilarity.”
- England (2021): “HML synthesis of distinguished processes.”
- Duong (2022): “Developing an educational tool for linear-time–branching-time spectroscopy.”
- Alshukairi (2022): “Automatisierte Reduktion von reaktiver zu starker Bisimilarität.”
- Adler (2022): “Simulation fehlertoleranter Konsensalgorithmen in Hash.ai.”
- Sandt (2022): “A video game about reactive bisimilarity.”
- Lönne (2023): “An educational computer game about counterfactual truth conditions.”
- Hauschild (2023): “Nonlinear counterfactuals in Isabelle/HOL.”
- Stöcker (2024): “Higher-order diadic μ -calculus—An efficient framework for checking process equivalences?”
- Kurzan (2024): “Implementierung eines Contrasimilarity-Checkers für mCRL2.”

This thesis itself. This document itself is created using Quarto (2025). Its HTML version is deployed to <https://generalized-equivalence-checking.equiv.io/>. One can find its source on <https://github.com/benkeks/generalized-equivalence-checking/>.

And of course, one can just continue reading it right here. In Chapter 2, we will dive into the formal description of equivalence and difference in program behavior, and start building a game framework that lets us solve *our* problem of generalized equivalence checking in one shot.

2 Preliminaries: Communicating Systems and Games

The core background of this thesis is the *trinity of characterizations for behavioral equivalences*: relational, modal, and game-theoretic.

This chapter takes a tour into the field of formalisms used to model programs and communicating systems, which are accompanied by many notions for behavioral equivalence and refinement to relate or distinguish their models. As core formalisms we will introduce transition systems and the Calculus of Communicating Systems (CCS) in Section 2.1.

The tour is *agnostic*, building on the basic formalism of *labeled transition systems* and standard equivalences such as *trace equivalence* and *bisimilarity*. Simultaneously, the approach is *opinionated*, focusing on Milner’s tradition of concurrency theory with a strong pull towards *game characterizations*.

Figure 2.1 shows the scaffold of this section along the trinity, instantiated to the classic notion of bisimilarity:

- Section 2.2 introduces (among other notions) bisimilarity through its *relational characterization* in terms of symmetric simulation relations.
- Section 2.3 treats the dual connection of bisimilarity and the distinguishing powers of a *modal logic*, known as the *Hennessey–Milner theorem*.
- Section 2.4 shows that both, the relations to validate bisimilarity and the modal formulas to falsify it, appear as certificates of strategies in a *reachability game* where an attacker tries to tell states apart and a defender tries to prevent this.

A reader familiar with the contents of Figure 2.1 might mostly skim through this chapter of preliminaries (although they are quite exciting!). This chapter aims to seed two core insights for the upcoming chapters:

i Idea 2: It’s all a game!

Equivalence games are a versatile way to handle behavioral equivalences and obtain decision procedures. The Hennessey–Milner theorem appears as a shadow of the determinacy of the bisimulation reachability

Related publications. This chapter is based on my introductory lectures and Isabelle/HOL theory on behavioral equivalences, given in the Master course “Research at Work” at TU Berlin in Winter 2023/24. We aim to be mostly compatible with the textbook *Reactive systems* (Aceto et al., 2007).

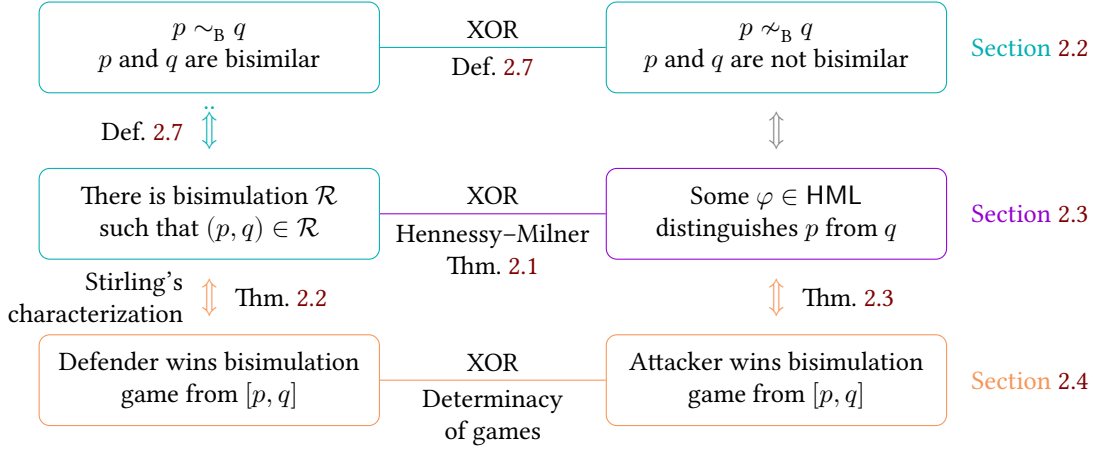


Figure 2.1: Core correlations for bisimilarity between relational definition, modal distinguishability, and equivalence game.

game.

We will move towards motivating the games through logics.

i Idea 3: Modal first!

Modal characterizations allow a uniform handling of the hierarchy of equivalences. Productions in grammars of potential distinguishing formulas translate to game rules for equivalence games.

Both points might seem non-standard to those more accustomed to relational or denotational definitions of behavioral equivalences. To provide them with a bridge, we will start with relational and denotational definitions—but once we have crossed this bridge, we will stay in the realm of games and logics.

2.1 Behavior of Programs

Every computer scientist has some model in their head of what it is that their algorithms and programs are doing. Usually these models are wrong,⁴ especially, once concurrency enters the picture. The area of *formal methods* tries to make models sufficiently precise that one, at least, can say what went wrong.

2.1.1 Labeled Transition Systems

Labeled transition systems are the standard formalism to discretely express the state space of programs, algorithms, and more. One can think of them as nondeterministic finite automata without finiteness constraint.

Definition 2.1 (Transition systems). A *transition system* $\mathcal{S} = (\mathcal{P}, \text{Act}, \rightarrow)$ consists of

- \mathcal{P} , a set of *states*,
- Act , a set of *actions*, and
- $\rightarrow \subseteq \mathcal{P} \times \text{Act} \times \mathcal{P}$, a *transition relation*.

We use infix notation for arrows, that is, $p \xrightarrow{\alpha} p'$ stands for $(p, \alpha, p') \in \rightarrow$. We write $\text{Der}(p, \alpha)$ for the set of *derivative states* $\{p' \mid p \xrightarrow{\alpha} p'\}$ and $\text{Ini}(p)$ for the set of enabled actions $\{\alpha \mid \exists p'. p \xrightarrow{\alpha} p'\}$. We sometimes lift transitions to sets $P, P' \subseteq \mathcal{P}$ with $P \xrightarrow{\alpha} P'$ iff $P' = \{p' \in \mathcal{P} \mid \exists p \in P. p \xrightarrow{\alpha} p'\}$.

There is a canonical example used to discuss equivalences within transition systems, which we want to draw from. We will take the formulation that Henzinger used at CAV'23 as seen in Figure 2.2.

Example 2.1 (A classic example). Consider the transition system $\mathcal{S}_{\text{PQ}} = (\{P, p_a, p_b, p_1, p_2, Q, q_{ab}, q_1, q_2\}, \{a, b, \tau\}, \rightarrow_{\text{PQ}})$ given by the following graph:

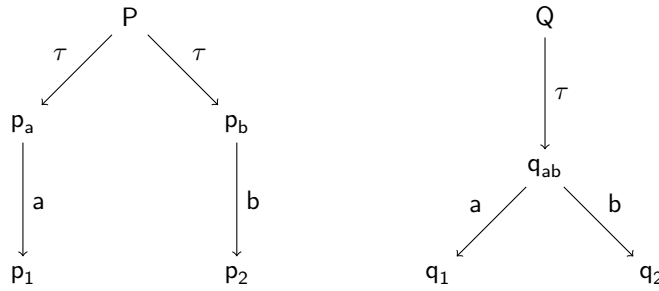


Figure 2.3: Example system \mathcal{S}_{PQ} .

The program described by the transitions from P chooses nondeterministically during a τ -step between two options and then offers only *either* a or b . The program of Q , on the other hand, performs a τ -step and then offers the choice between options a and b to the environment.

There are two things one might wonder about Example 2.1:

1. Should one care about nondeterminism in programs? Section 2.1.2 shows how nondeterminism arises naturally in concurrent programs. And, of course, many specifications leave *undefined behavior*.
2. Should one consider P and Q equivalent? This heavily depends. Section 2.2 will introduce a notion of equivalence under which the two are equivalent and one under which they differ.

Remark 2.1 (A note on τ). The action τ (the Greek letter “tau”) will stand for *internal behavior* in later chapters and receive special treatment in Chapter 6.

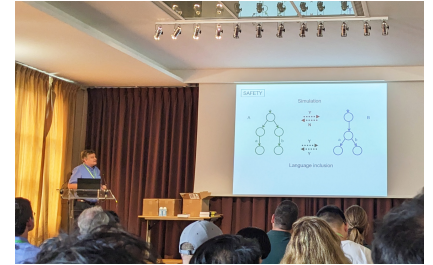


Figure 2.2: Tom Henzinger employing Example 2.1 during CAV'23 to stress that “games are everywhere.” (Incidentally, this chapter follows a similar mission.)

For the scope of this chapter and the following three, τ is an action like every other.

Generally, this thesis aims to be consistent with notation and naming in surrounding literature. For the internal action, the whole field has converged to τ in italics—so, we will run with this. Otherwise, we follow the convention to write literals and constant names in sans-serif and variables in *italics*.

2.1.2 Calculus of Communicating Systems

To talk about programs in this thesis, we use Milner’s (1980) *Calculus of Communicating Systems* (CCS), which—together with other great contributions—earned him the Turing award. It is a tiny concurrent programming language that fits in your pocket, and can be properly described mathematically!⁵

Definition 2.2 (Syntax of CCS). Let \mathcal{C} be a set of channel names, and \mathcal{X} a set of process names. Then, CCS is the set of processes given by the following grammar:⁶

$P ::=$	$c.P$	with $c \in \mathcal{C}$	“input action prefix”
	$\bar{c}.P$	with $c \in \mathcal{C}$	“output action prefix”
	$\tau.P$		“internal action”
	0		“null process”
	X	with $X \in \mathcal{X}$	“recursion”
	$P + P$		“choice”
	$P \mid P$		“parallel composition”
	$P \setminus A$	with $A \subseteq \mathcal{C}$	“restriction”

Intuitively, input action c represents receiving and output action \bar{c} expresses sending on channel $c \in \mathcal{C}$. A pair of input and output can “react” in a communication situation and only become internal activity τ in the view of the environment. Taken together, CCS processes exhibit the actions $\text{Act}_{\text{CCS}} := \mathcal{C} \cup \{\bar{c} \mid c \in \mathcal{C}\} \cup \{\tau\}$.

Each part of the syntax tree must end in a 0 -process or recursion. For brevity, we usually drop a terminal 0 when writing terms, e.g., just writing ac for $ac.0$.

We place parenthesis (...) in terms where the syntax trees are otherwise ambiguous, but understand the choice operator $+$ and the parallel operator \mid to be associative. This convention means that $(P_1 + P_2) + P_3$ and $P_1 + (P_2 + P_3)$ are representations of the same term in our view.

Example 2.2 (Concurrent philosophers). Following tradition, we will express our examples in terms of philosophers who need forks to eat spaghetti.⁷ Consider two philosophers P_A and P_B (from Figure 2.4) who want to grab a resource fork, in order to eat, which we model as receiving communication. We express P_A eating with a and P_B eating with b . The philosopher processes

⁵ To those not familiar with CCS: The following exposition of CCS is quite condensed. Gentler introduction can be found in textbooks like Milner (1989) or Aceto et al. (2007) and in the *interactive learning experience* <https://book.pseuco.com/>.

⁶ To those familiar with CCS dialects: For the examples of the thesis, a simple CCS variant without value passing and relabeling suffices.

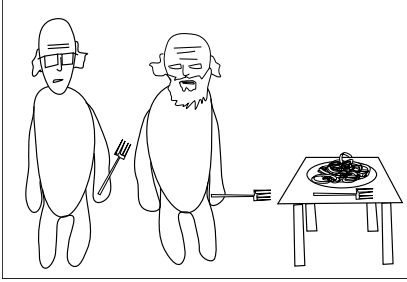


Figure 2.4: Scenario of two philosophers needing a (second) fork to eat pasta.

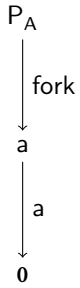


Figure 2.5: Philosopher P_A , on their own.

⁷ The philosopher meme is due to Dijkstra and Hoare (1985). Of course, you can just as well read the examples to be about computer programs that race for resources.

read:

$$P_A := \text{fork.a.0}$$

$$P_B := \text{fork.b.0}$$

A transition system representation of P_A 's behavior can be seen in Figure 2.5. Process P captures the whole scenario where the two philosophers compete for the fork using communication:

$$P := (\overline{\text{fork.0}} \mid P_A \mid P_B) \setminus \{\text{fork}\}$$

The restriction $\dots \setminus \{\text{fork}\}$ expresses that the fork-channel can only be used for communication *within* the system.

As the $\overline{\text{fork}}$ -message can be consumed by just one of the two philosophers, process P expresses exactly the program behavior seen in state P of Example 2.1. A process matching Q will follow in Example 2.3.

The formal relationship between process terms of CCS and their transition system is established by the following semantics.

Definition 2.3 (CCS semantics). Given an assignment of names to processes, $\mathcal{V}: \mathcal{X} \rightarrow \text{CCS}$, the operational semantics $\rightarrow_{\text{CCS}} \subseteq \text{CCS} \times \text{Act}_{\text{CCS}} \times \text{CCS}$ is defined inductively by the inference rules:

$$\begin{array}{c} \text{Pre} \frac{}{\alpha.P \xrightarrow{\alpha}_{\text{CCS}} P} \quad \text{Rec} \frac{P \xrightarrow{\alpha}_{\text{CCS}} P' \quad \mathcal{V}(X) = P}{X \xrightarrow{\alpha}_{\text{CCS}} P'} \\ \\ \text{Choice}_1 \frac{P_1 \xrightarrow{\alpha}_{\text{CCS}} P'_1}{P_1 + P_2 \xrightarrow{\alpha}_{\text{CCS}} P'_1} \quad \text{Choice}_2 \frac{P_2 \xrightarrow{\alpha}_{\text{CCS}} P'_2}{P_1 + P_2 \xrightarrow{\alpha}_{\text{CCS}} P'_2} \\ \\ \text{Par}_1 \frac{P_1 \xrightarrow{\alpha}_{\text{CCS}} P'_1}{P_1 \mid P_2 \xrightarrow{\alpha}_{\text{CCS}} P'_1 \mid P_2} \quad \text{Par}_2 \frac{P_2 \xrightarrow{\alpha}_{\text{CCS}} P'_2}{P_1 \mid P_2 \xrightarrow{\alpha}_{\text{CCS}} P_1 \mid P'_2} \\ \\ \text{Com}_1 \frac{P_1 \xrightarrow{c}_{\text{CCS}} P'_1 \quad P_2 \xrightarrow{\bar{c}}_{\text{CCS}} P'_2}{P_1 \mid P_2 \xrightarrow{\tau}_{\text{CCS}} P'_1 \mid P'_2} \quad \text{Com}_2 \frac{P_1 \xrightarrow{\bar{c}}_{\text{CCS}} P'_1 \quad P_2 \xrightarrow{c}_{\text{CCS}} P'_2}{P_1 \mid P_2 \xrightarrow{\tau}_{\text{CCS}} P'_1 \mid P'_2} \\ \\ \text{Res} \frac{P \xrightarrow{\alpha}_{\text{CCS}} P' \quad \nexists c \in A. \alpha = c \vee \alpha = \bar{c}}{P \setminus A \xrightarrow{\alpha}_{\text{CCS}} P' \setminus A} \end{array}$$

A process $P \in \text{CCS}$ now denotes a node within the labeled transition system $(\text{CCS}, \text{Act}_{\text{CCS}}, \rightarrow_{\text{CCS}})$.⁸

So, when we were writing “ $P_A := \text{fork.a.0}$ ” above, this was actually to claim that we are talking about a CCS system where the value of \mathcal{V} for the *process name* $P_A \in \mathcal{X}$ is defined by $\mathcal{V}(P_A) := \text{fork.a.0}$. By the semantics, this also leads to the existence of a *state* P_A in the CCS transition system, and usually that is the entity we are interested in when defining a process.

Feel free to go ahead and check that the transitions of Example 2.1 indeed match those that Definition 2.3 prescribes for P of Example 2.2! (For readability, Example 2.1, has shorter state names, however.) For instance, the

⁸ Such rules are read like this: “If we can derive the facts above the line, we may infer the relation below.” Rule Pre, having no premises, serves as the only axiom of this inference system. Figure 2.6 shows how to stack such rules to infer that a certain step is possible for a process.

$$\begin{array}{c}
\text{Pre} \xrightarrow{\quad} \text{Rec} \xrightarrow{\text{fork}.a \xrightarrow{\text{fork}}_{\text{CCS}} a} \mathcal{V}(P_A) = \text{fork}.a \\
\text{Pre} \xrightarrow{\quad} \text{Par}_1 \xrightarrow{P_A \xrightarrow{\text{fork}}_{\text{CCS}} a} \\
\text{Com}_2 \xrightarrow{\overline{\text{fork}} \xrightarrow{\text{fork}}_{\text{CCS}} 0} P_A \mid P_B \xrightarrow{\text{fork}}_{\text{CCS}} a \mid P_B \\
\text{Res} \xrightarrow{\overline{\text{fork}} \mid P_A \mid P_B \xrightarrow{\tau}_{\text{CCS}} 0 \mid a \mid P_B} \tau \notin \{\text{fork}\} \\
\text{Rec} \xrightarrow{(\overline{\text{fork}} \mid P_A \mid P_B) \setminus \{\text{fork}\} \xrightarrow{\tau}_{\text{CCS}} (0 \mid a \mid P_B) \setminus \{\text{fork}\}} \mathcal{V}(P) = (\overline{\text{fork}} \mid P_A \mid P_B) \setminus \{\text{fork}\} \\
P \xrightarrow{\tau}_{\text{CCS}} (0 \mid a \mid P_B) \setminus \{\text{fork}\}
\end{array}$$

Figure 2.6: Deriving CCS transitions due to communication.

transition $P \xrightarrow{\tau} p_a$ in the CCS system of Example 2.1 would be justified by the derivation in Figure 2.6 with $p_a = (0 \mid a \mid P_B) \setminus \{\text{fork}\}$.

Therefore, after one step from P , the (internal) fork has been consumed, and only one philosopher may eat. From the outside, this appears as nondeterminism of the system.

Nondeterminism as in P of Example 2.1 can be understood as a natural phenomenon in models with concurrency. The model leaves unspecified which of two processes will consume an internal resource and, to the outsider, it is transparent which one took the resource until they communicate. There are other ways how nondeterminism plays a crucial role in models, for instance, as consequence of abstraction or of parts that are left open in specifications.

The second process Q of Example 2.1 can be understood as a *deterministic* sibling of P .

Example 2.3 (Deterministic philosophers). A process matching the transitions from Q in Example 2.1 would be the following, where the philosophers take the fork as a team and then let the environment choose who of them eats:

$$Q := (\overline{\text{fork}}.0 \mid \text{fork}.(a.0 + b.0)) \setminus \{\text{fork}\}.$$

But are P and Q *equivalent*?

2.2 Behavioral Equivalences

A behavioral equivalence formally defines when to consider two processes (or states, or programs) as *equivalent*. Evidently, there are different ways of choosing such a notion of equivalence. Also, sometimes we are interested in a behavioral *preorder*, for instance, as a way of saying that a program does “less than” what some specification allows. Such a relationship is often also called *refinement* or *inclusion*, but we will go with *preorder*.

This section quickly introduces the most common representatives of behavioral equivalences: Trace equivalence (and preorder), simulation equivalence (and preorder), and bisimilarity. We will then observe that the notions themselves can be compared in a hierarchy of equivalences.

2.2.1 Trace Equivalence

Every computer science curriculum features automata and their languages sometime at the beginning. Accordingly, comparing two programs in terms of the sequences of input/output events they might expose is a natural starting point to talk about behavioral equivalences. Such sequences of actions are referred to as *traces*.

Definition 2.4 (Traces). The set of traces of a state $\text{Traces}(p) \subseteq \text{Act}^*$ is inductively defined as

- $() \in \text{Traces}(p)$,⁹
- $\alpha\vec{w} \in \text{Traces}(p)$ if there is p' with $p \xrightarrow{\alpha} p'$ and $\vec{w} \in \text{Traces}(p')$.¹⁰

Definition 2.5 (Trace equivalence). Two states p and q are considered *trace-equivalent*, written $p \sim_T q$, if $\text{Traces}(p) = \text{Traces}(q)$.¹¹

States are *trace-preordered*, $p \preceq_T q$, if $\text{Traces}(p) \subseteq \text{Traces}(q)$.¹²

Example 2.4. The traces for the processes of Example 2.1 would be $\text{Traces}(P) = \{(), \tau, \tau a, \tau b\} = \text{Traces}(Q)$. Consequently, P and Q are trace-equivalent, $P \sim_T Q$.

As $\text{Traces}(p_a) = \{(), a\} \subseteq \{(), a, b\} = \text{Traces}(q_{ab})$, p_a is trace-preordered to q_{ab} , $p_a \preceq_T q_{ab}$. This ordering is strict, that is, $q_{ab} \not\preceq_T p_a$, due to $b \in \text{Traces}(q_{ab})$ but $b \notin \text{Traces}(p_a)$. We could say that trace b constitutes a *difference* between q_{ab} and p_a .

Proposition 2.1. *Trace preorder \preceq_T is indeed a preorder (i.e., transitive and reflexive)¹³ and trace equivalence \sim_T is indeed an equivalence relation¹⁴ (i.e., transitive, reflexive, and moreover symmetric).¹⁵*

Trace equivalence (and preorder) give straightforward *denotational semantics* to programs: The sets of traces they might expose. For many formal languages, these mathematical objects can be constructed directly from the expressions of the language. With the idea that the program text “denotes” its possible executions, the set of traces is called a “denotation” in this context. CCS, as we use it, follows another approach to semantics, namely the one of “operational semantics,” where the meaning of a program is in how it might interact.


There are several reasons why computer scientists did not content themselves with trace equivalence when studying interactive systems. The core argument is that, in this context, one usually does not want to consider processes like P and Q to be equivalent: The two might interact differently with an environment.


⁹ We denote the empty word by $()$.


¹⁰  abbreviation [Labeled_Transition_Systems.lts.traces](#)

¹¹  abbreviation [Strong_Equivalences.lts.trace_equivalent](#)

¹²  definition [Strong_Equivalences.lts.trace_preordered](#)

¹³  lemma [Strong_Equivalences.lts.trace_preorder_transitive](#)

¹⁴  lemma [Strong_Equivalences.lts.trace_equivalence_equiv](#)

¹⁵ This thesis states many facts, only linking to their proof. In this case, proofs can be found in the Isabelle/HOL formalization, indicated by .

Example 2.5 (Live lecture). Consider a context in which A actually does not want to eat and the event of B eating is followed by B giving a live lecture:

$$(\dots \mid \overline{b}.\overline{\text{lecture}}.0) \setminus \{a, b\}$$

If we insert Q for the “...” in this context, the choice after the collective fork-acquisition is resolved to the B-path. So the overall system chooses for B to eat, and the lecture happens:

$$(Q \mid \overline{b}.\overline{\text{lecture}}.0) \setminus \{a, b\} \xrightarrow{\tau} \cdot \xrightarrow{\tau} \cdot \xrightarrow{\overline{\text{lecture}}} (0 \mid 0) \setminus \{a, b\}$$

With P, an analogous path is possible. But there is, moreover, a possibility of A obtaining the fork, which prevents B from eating. There, the system can end up deadlocked with no lecture:

$$(P \mid \overline{b}.\overline{\text{lecture}}.0) \setminus \{a, b\} \xrightarrow{\tau} ((0 \mid a \mid P_B) \setminus \{\text{fork}\} \mid \overline{b}.\overline{\text{lecture}}.0) \setminus \{a, b\} \nrightarrow$$

So, if B’s teaching actually happening matters to us, we might say that scenario P and Q are not quite equivalent.¹⁶

Computer scientists might not all care about philosophy lectures. But they care about the *liveness* of interactive programs. Therefore, they have invented many program equivalences that pay more attention to *possibilities coming and going*. The most prominent ones are the topic of the next subsection.

2.2.2 Similarity and Bisimilarity

The other big approach to behavioral equivalence of programs is the one of relating parts of their state spaces to one-another. The idea here is to identify which states of one program can be used to *simulate* the behavior of the other.

Definition 2.6 (Simulation). A relation on states, $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$, is called a *simulation* if, for each $(p, q) \in \mathcal{R}$ and $\alpha \in \text{Act}$ with $p \xrightarrow{\alpha} p'$ there is a q' with $q \xrightarrow{\alpha} q'$ and $(p', q') \in \mathcal{R}$.¹⁷

Definition 2.7 ((Bi-)similarity). Simulation preorder, simulation equivalence and bisimilarity are defined as follows:

- p is *simulated by* q , $p \preceq_S q$, iff there is a simulation \mathcal{R} with $(p, q) \in \mathcal{R}$.¹⁸
- p is *similar to* q , $p \sim_S q$, iff $p \preceq_S q$ and $q \preceq_S p$.¹⁹
- p is *bisimilar to* q , $p \sim_B q$, iff there is a *symmetric* simulation \mathcal{R} (i.e. $\mathcal{R} = \mathcal{R}^{-1}$) with $(p, q) \in \mathcal{R}$.²⁰

We also call a symmetric simulation *bisimulation* for short.²¹ Canceled symbols of relations refer to their negations, for instance, $p \not\preceq_S q$ iff there is *no* simulation \mathcal{R} with $(p, q) \in \mathcal{R}$.

Example 2.6 (Philosophical simulations). The following relations are simulations on the system of Example 2.1:

¹⁶ Traditionally, this feature of trace equivalence is often framed in terms of vending machines. On <https://book.pseuco.com/#/interactive/trace-equivalent-vending-machines>, you can play through a comparable scenario of ordering drinks where *you are the environment*.

¹⁷ definition [Strong_Equivalences.Its.simulation](#)

¹⁸ definition [Strong_Equivalences.Its.simulated_by](#)

¹⁹ abbreviation [Strong_Equivalences.Its.similar](#)

²⁰ definition [Strong_Equivalences.Its.bisimilar](#)

²¹ Other authors use a weaker definition, namely, that \mathcal{R} is a bisimulation if \mathcal{R} and \mathcal{R}^{-1} are simulations. Both definitions lead to the characterization of the same notion of bisimilarity.

- the empty relation $\mathcal{R}_\emptyset := \emptyset$;
- the identity relation $\mathcal{R}_{\text{id}} := \text{id}_{\{P, P_a, P_b, P_1, P_2, Q, Q_{ab}, Q_1, Q_2\}} = \{(P, P), (P_a, P_a), (P_b, P_b), (P_1, P_1), (P_2, P_2), (Q, Q), (Q_{ab}, Q_{ab}), (Q_1, Q_1), (Q_2, Q_2)\}$;
- the universal relation between all terminal states
 $\mathcal{R}_{\text{term}} := \{P_1, P_2, Q_1, Q_2\} \times \{P_1, P_2, Q_1, Q_2\}$;
- more generally, the relation from terminal states to all other states:
 $\mathcal{R}_{\text{up}} := \{P_1, P_2, Q_1, Q_2\} \times \mathcal{P}$;
- a minimal simulation for P and Q:
 $\mathcal{R}_{PQ} := \{(P, Q), (P_a, Q_{ab}), (P_b, Q_{ab}), (P_1, Q_1), (P_2, Q_2)\}$;
- and the combination of the above $\mathcal{R}_{\text{big}} := \mathcal{R}_{\text{id}} \cup \mathcal{R}_{\text{term}} \cup \mathcal{R}_{\text{up}} \cup \mathcal{R}_{PQ}$.

The simulation \mathcal{R}_{PQ} shows that $P \preceq_S Q$.

However, there is no simulation that preorders Q to P, as there is no way to simulate the transition $Q \xrightarrow{\tau} q_{ab}$ from P for lack of a successor that allows a and b as does q_{ab} . (In Section 2.3, we will discuss how to capture such differences more formally.)

Thus, $Q \not\preceq_S P$, and $P \sim_S Q$. Moreover, there cannot be a symmetric simulation, $P \sim_B Q$.

Proposition 2.2. *The simulation preorder \preceq_S is indeed a preorder²², and \sim_S and \sim_B are equivalences.²³*

Example 2.6 shows that similarity and bisimilarity do not imply trace equivalence. Still, the notions are connected.

2.2.3 Equivalence Hierarchies

Bisimilarity, similarity and trace equivalence form a small hierarchy of equivalences in the sense that they imply one-another in one direction, but not in the other. Let us quickly make this formal:

Lemma 2.1. *The relation \sim_B is itself a symmetric simulation.²⁴*

Corollary 2.1. *If $p \sim_B q$, then $p \sim_S q$.²⁵*

Lemma 2.2. *If $p \preceq_S q$, then $p \preceq_T q$.²⁶ (Consequently, $p \sim_S q$ also implies $p \sim_T q$.²⁷)*

We also have seen with Example 2.6 that this hierarchy of implications (visualized in Figure 2.7) is strict between trace and simulation preorder in the sense that there exist p, q with $p \preceq_T q$ but not $p \preceq_S q$. The hierarchy also is strict between similarity and bisimilarity as the following example shows.

Example 2.7 (Trolled philosophers). Let us extend Q of Example 2.3 to include a troll process (highlighted in blue) that might consume the fork and then do nothing:

$$T := (\overline{\text{fork}} \mid \text{fork} \mid \text{fork} \cdot (a + b)) \setminus \{\text{fork}\}.$$

This adds another deadlock state to the transition system, seen in Figure 2.8.

²² lemma Strong_Equivalences.lts.simulation_preorder_transitive
²³ lemma Strong_Equivalences.lts.bisimilarity_equiv

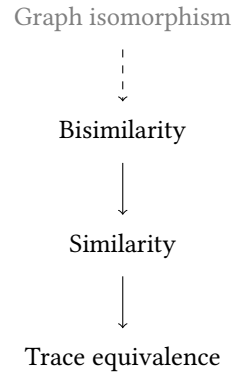


Figure 2.7: Core hierarchy of equivalences. (Arrows mean implication.)

²⁴ lemma Strong_Equivalences.lts.bisim_bisim
²⁵ lemma Strong_Equivalences.lts.bisim_bisim
²⁶ lemma Strong_Equivalences.lts.sim_implies_trace_preord
²⁷ lemma Strong_Equivalences.lts.sim_eq_implies_trace_eq

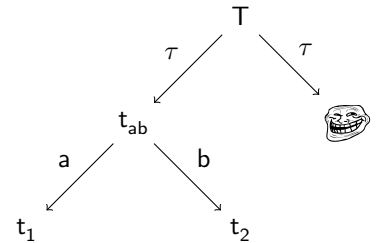


Figure 2.8: Example with new deadlock.

With respect to similarity, this change is invisible, that is, $Q \sim_S T$. (Formal argument: The relation $\{(Q, T), (T, Q), (q_{ab}, t_{ab}), (t_{ab}, q_{ab}), (\text{deadlock}, q_{ab})\} \cup \{q_1, q_2, t_1, t_2, \text{deadlock}\} \times \{q_1, q_2, t_1, t_2, \text{deadlock}\}$ is a simulation.)

However, to bisimilarity, $T \xrightarrow{\tau} \text{deadlock}$ constitutes a difference. There cannot be a *symmetric* simulation handling this transition as Q has no deadlocked successors. Thus $Q \not\sim_B T$.

The equivalences we have discussed so far could also be understood as abstractions of an even finer equivalence, namely graph isomorphism:

Definition 2.8 (Graph isomorphism). A bijective function $f: \mathcal{P} \rightarrow \mathcal{P}$ is called a *graph isomorphism* on a transition system if, for all p, p', α , the transition $p \xrightarrow{\alpha} p'$ exists if and only if the transition $f(p) \xrightarrow{\alpha} f(p')$ exists.²⁸

Two states p and q are considered *graph-isomorphism-equivalent*, $p \sim_{\text{ISO}} q$, iff there is a graph isomorphism f with $f(p) = q$.²⁹

Example 2.8. Consider the transition system in Figure 2.9. $p_{\text{even}} \sim_{\text{ISO}} p_{\text{odd}}$ because $f := \{p_{\text{even}} \mapsto p_{\text{odd}}, p_{\text{odd}} \mapsto p_{\text{even}}\}$ is a graph isomorphism.

Lemma 2.3. The relation \sim_{ISO} is itself a symmetric simulation and thus $p \sim_{\text{ISO}} q$ implies $p \sim_B q$.³⁰

Once again, the hierarchy is strict because of bisimilarity being less restricted in the matching of equivalent states.

Example 2.9 (Graph isomorphism counts too much). Consider the processes $P_1 := (\text{fork} \mid \text{fork}) \setminus \{\text{fork}\}$ and $P_2 := (\text{fork} \mid \text{fork} \mid \text{fork}) \setminus \{\text{fork}\}$. P_1 can transition to $(0 \mid 0) \setminus \{\text{fork}\}$, while P_2 has two options, namely $(0 \mid 0 \mid \text{fork}) \setminus \{\text{fork}\}$ and $(0 \mid \text{fork} \mid 0) \setminus \{\text{fork}\}$. All three reachable processes are deadlocks and thus isomorphic. But $P_1 \not\sim_{\text{ISO}} P_2$ because no bijection can connect the one successor of P_1 and the two of P_2 . However, $P_1 \sim_B P_2$, as bisimilarity is more relaxed.

Graph isomorphism is the strongest equivalence, we have discussed so far. But stronger needs not be better.

2.2.4 Congruences

One of the prime quality criteria for behavioral equivalences is whether they form *congruences* with respect to fitting semantics or other important transformations. A congruence relates mathematical objects that can stand in for one-another in certain contexts, which, for instance, allows term rewriting. The concept is closely linked to another core notion of mathematics: monotonicity.

Definition 2.9 (Monotonicity and congruence). An n -ary function $f: B_1 \times \dots \times B_n \rightarrow C$ is called *monotonic* with respect to a family of preorders (B_k, \leq_k) and (C, \leq) iff, for all $b \in B_1 \times \dots \times B_n$ and $b' \in B_1 \times \dots \times B_n$, it is the case that $b_k \leq_k b'_k$ for all $k \leq n$ implies that $f(b) \leq f(b')$. We will usually

²⁸ definition Strong_Equivalences.Its.isomorphism

²⁹ definition Strong_Equivalences.Its.is_isomorphic_to

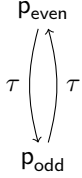


Figure 2.9: Transition system with an isomorphic cycle.

³⁰ lemma Strong_Equivalences.Its.iso_implies_bisim

encounter settings where all components use the same order $(B_1, \leq_1) = \dots = (B_n, \leq_n) = (C, \leq)$.

The relation \leq is then referred to as a *precongruence* for f . If \leq moreover is symmetric (and thus an equivalence relation), then \leq is called a *congruence* for f .

Example 2.10 (Parity as congruence). As a standard example for a congruence, consider the equivalence relation of equally odd numbers:

$$\sim_{\text{odd}} := \{(m, n) \in \mathbb{N} \times \mathbb{N} \mid m \bmod 2 = n \bmod 2\}.$$

For instance, $1 \sim_{\text{odd}} 3$ and $0 \sim_{\text{odd}} 42$, but not $42 \sim_{\text{odd}} 23$.

\sim_{odd} is a congruence for addition and multiplication. For instance, the sum of two odd numbers will always be even; the product of two odd numbers, will always be odd.

But \sim_{odd} is no congruence for integer division. For instance, $2 \sim_{\text{odd}} 4$, but $2/2 = 1 \not\sim_{\text{odd}} 2 = 4/2$.

Proposition 2.3. *Trace equivalence and bisimilarity on the CCS transition system (Definition 2.3) are congruences for the operators of CCS (Definition 2.2).*³¹

The nice thing about congruences is that we can use them to *calculate with terms*, as is common in mathematics.

Graph isomorphism fails to be a congruence for CCS operators. For instance, consider $a.(0 \mid 0) \sim_{\text{ISO}} a.0$. But if one inserts the two terms in a choice context, the results differ with respect to isomorphism: $a.0 + a.(0 \mid 0) \not\sim_{\text{ISO}} a.0 + a.0 \sim_{\text{ISO}} a.0$.

By now, we have encountered some behavioral equivalences. Both, bisimilarity and trace equivalence make for fine congruences that allow calculations on process-algebraic expressions of CCS. Which one to choose? That there is no definitive answer is a root cause of this thesis.

Remark 2.2 (Origins of linear time and branching time). This thesis is a consequence of the classic idea that behavioral equivalences form a *spectrum* between linear-time notions and branching-time notions. This hierarchy corresponds to “how much” calculation is allowed on algebraic expressions of processes.

Bisimilarity is the prototypical *branching-time* equivalence: every decision during execution counts. Trace equivalence is the prototypical *linear-time* notion: only sequences of actions count. Therefore, more can be seen as equivalent than in bisimilarity.

In the concurrency theory community, two prototypical *process algebras* stand for this spread: CCS for bisimilarity and branching time, and CSP for trace equivalence and linear-time. CSP (“Communicating Sequential Processes”) is due to Hoare, another Turing award winner.

Hoare (1985, Section 7.4.1) summarizes the opposing design philosophies between CCS and CSP as follows:

³¹ For a generic proof of the congruence properties of trace equivalence, bisimilarity, and most other notions of the equivalence spectrum, see Gazda et al. (2020).

[...] CCS is intended to serve as a framework for a family of models, each of which may make more identifications than CCS but cannot make less. To avoid restricting the range of models, CCS makes only those identifications which seem absolutely essential. In the mathematical model of [CSP] we have pursued exactly the opposite goal—we have made as many identifications as possible, preserving only the most essential distinctions. We have therefore a far richer set of algebraic laws.

The first comprehensive study on the correspondence between the hierarchy of equivalences and algebraic laws has been dubbed “linear-time–branching-time spectrum” by van Glabbeek (1990).³²

Historically, therefore, this thesis originates from the tension between the works of two Turing award winners: Milner’s CCS and Hoare’s CSP. Or, as Hoare (2006) puts it: “It seems that CCS and CSP occupy extreme ends of almost every spectrum.”

But do not worry—although our research question is a consequence of the possibility to come up with various process algebras, this thesis will stick to only one, CCS.

2.2.5 Quotient Systems and Minimizations

One of the prime applications of behavioral equivalences is to battle the *state space explosion problem*: The transition systems of parallel processes quickly grow in size, usually exponentially with regard to the number of communication participants. But many of the states tend to be equivalent in behavior, and can be treated *as one* with their equals. Such *minimization* enables the handling of vastly bigger input models.

Example 2.11 (Trolled philosophers, minimized). In Example 2.7 of “trolled philosophers,” all terminal states are bisimilar, $t_1 \sim_B t_2 \sim_B \text{Ⓜ}$. We can thus merge them into one state Ⓜ^m as depicted in Figure 2.10, without affecting the behavioral properties of the other states with respect to bisimilarity, that is, $t_{ab} \sim_B t_{ab}^m$ and $T \sim_B T^m$.

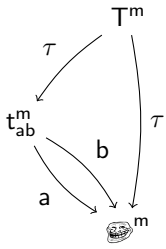


Figure 2.10: Minimized version of Figure 2.8.

Definition 2.10 (Quotient systems). Given an equivalence relation \sim on states, and a transition system $\mathcal{S} = (\mathcal{P}, \text{Act}, \rightarrow)$, each *equivalence class* for $p \in \mathcal{P}$ is defined $[p]_\sim := \{p' \in \mathcal{P} \mid p \sim p'\}$.

The *quotient system* is defined as $\mathcal{S}/\sim := (\mathcal{P}/\sim, \text{Act}, \rightarrow_\sim)$, where \mathcal{P}/\sim is the set of equivalence classes $\{[p]_\sim \mid p \in \mathcal{P}\}$, and

$$P \xrightarrow{\alpha}_\sim P' \text{ iff there are } p \in P \text{ and } p' \in P' \text{ such that } p \xrightarrow{\alpha} p'.$$

Proposition 2.4. On the combined system of \mathcal{S} and \mathcal{S}/\sim_B^S , states and their derived class are bisimilar, $p \sim_B [p]_{\sim_B^S}$.

The same reasoning does not apply to graph isomorphism: In Example 2.11, the terminal states might be graph-isomorphic, but merging them changes the number of states and thus prevents isomorphism between T and T^m .

Together with the issue of congruence, we have now seen two reasons why graph isomorphism does not allow the kind of handling we hope for in behavioral equivalences. Thus, the following will focus on equivalences of bisimilarity and below. The next section will revisit a deeper, philosophical reason why bisimilarity is a reasonable limit of the properties one might care about in the comparison of processes.

2.3 Modal Logics

Modal logics are logics in which one can formalize how facts in one possible world relate to other possible worlds. In the computer science interpretation, the possible worlds are *program states*, and typical statements have the form: “If X happens during the execution, then Y will happen in the next step.”


Each modal logic naturally characterizes a behavioral equivalence. In this section, we show how such characterization works and argue that, for our purposes of comparative semantics, modal characterizations are a superb medium. In the next chapter, the equivalence hierarchy will turn out to be a hierarchy of modal sublogics.

2.3.1 Hennessy–Milner Logic to Express Observations

Hennessy & Milner (1980) introduce the modal logic that is now commonly called *Hennessy–Milner logic* (HML) as a “little language for talking about programs.” The idea is that HML formulas express “experiments” or “tests” that an observer performs interacting with a program.

Definition 2.11 (Hennessy–Milner logic). Formulas of *Hennessy–Milner logic* HML are given by the grammar:³³

$$\begin{array}{lll} \varphi & ::= & \langle \alpha \rangle \varphi \quad \text{with } \alpha \in \text{Act} \quad \text{“observation”} \\ & | & \bigwedge_{i \in I} \varphi_i \quad \text{with index set } I \quad \text{“conjunction”} \\ & | & \neg \varphi \quad \text{“negation”} \end{array}$$

³³  datatype [Hennessy_Milner_Logic.hml_formula](#)

We also write conjunctions as sets $\bigwedge \{\varphi_1, \varphi_2, \dots\}$. The empty conjunction $\bigwedge \emptyset$ is denoted by \top and serves as the nil-element of HML syntax trees. We also usually omit them when writing down formulas, e.g., shortening $\langle a \rangle \langle b \rangle \top$ to $\langle a \rangle \langle b \rangle$, which says that one may observe a and then b .

We will assume a form of associativity through the *convention that conjunctions are flat* in the sense that they do not immediately contain other conjunctions. Accordingly, we consider $\bigwedge \{\langle a \rangle, \bigwedge \{\langle b \rangle\}\}$ and $\bigwedge \{\langle a \rangle, \bigwedge \{\langle b \rangle\}, \top\}$ just as different representatives of the flattened formula $\bigwedge \{\langle a \rangle, \langle b \rangle\}$, stating that one may observe a *and* one may observe b .

We do not restrict the size of index sets in conjunctions. In particular, infinitary conjunctions are allowed, as we will discuss in Remark 2.3.

The intuition behind HML is that it describes *what sequences of behavior* one may or may not *observe* of a system. Observations $\langle \alpha \rangle \dots$ are used to build up possible action sequences; conjunctions $\bigwedge \{ \dots \}$ capture branching points in decision trees; and negations $\neg \dots$ describe impossibilities.

Definition 2.12 (HML semantics). Given a transition system $(\mathcal{P}, \text{Act}, \rightarrow)$, the semantics of HML $\llbracket \cdot \rrbracket : \text{HML} \rightarrow 2^{\mathcal{P}}$ is defined recursively by:³⁴

$$\begin{aligned} \llbracket \langle \alpha \rangle \varphi \rrbracket &:= \{ p \mid \exists p' \in \llbracket \varphi \rrbracket. p \xrightarrow{\alpha} p' \} \\ \llbracket \bigwedge_{i \in I} \varphi_i \rrbracket &:= \bigcap_{i \in I} \llbracket \varphi_i \rrbracket \\ \llbracket \neg \varphi \rrbracket &:= \mathcal{P} \setminus \llbracket \varphi \rrbracket \end{aligned}$$

Example 2.12. Let us consider some observations on the system of Example 2.1.

- $\llbracket \langle \tau \rangle \langle a \rangle \rrbracket = \{ P, Q \}$ as both, P and Q, expose the trace τa ,
- $Q \in \llbracket \langle \tau \rangle \bigwedge \{ \langle a \rangle, \langle b \rangle \} \rrbracket$, but $P \notin \llbracket \langle \tau \rangle \bigwedge \{ \langle a \rangle, \langle b \rangle \} \rrbracket$ as Q leaves the choice between a and b to the environment.
- $P \in \llbracket \langle \tau \rangle \neg \langle a \rangle \rrbracket$, but $Q \notin \llbracket \langle \tau \rangle \neg \langle a \rangle \rrbracket$ as P can internally decide against a.

2.3.2 Characterizing Bisimilarity via HML

We can now add the middle layer of our overview graphic in Figure 2.1: That two states are bisimilar precisely if they cannot be told apart using HML formulas.

Definition 2.13 (Distinctions and equivalences). We say that formula $\varphi \in \text{HML}$ *distinguishes* state p from state q if $p \in \llbracket \varphi \rrbracket$ but $q \notin \llbracket \varphi \rrbracket$.³⁵

We say a sublogic $\mathcal{O} \subseteq \text{HML}$ *preorders* state p to q , $p \preceq_{\mathcal{O}} q$, if no $\varphi \in \mathcal{O}$ distinguishes p from q .³⁶ If the preordering goes in both directions, we say that p and q are *equivalent* with respect to sublogic \mathcal{O} , written $p \sim_{\mathcal{O}} q$.³⁷


By this account, $\langle \tau \rangle \neg \langle a \rangle$ of Example 2.12 distinguishes P from Q. On the other hand, $\langle \tau \rangle \bigwedge \{ \langle a \rangle, \langle b \rangle \}$ distinguishes Q from P. (The direction matters!) For instance, the sublogic $\{ \langle \tau \rangle \langle a \rangle, \langle \tau \rangle \langle b \rangle \}$ preorders P and Q in both directions; so the two states are equivalent with respect to this logic.

Proposition 2.5 (Transitivity for free). *Consider an arbitrary HML sublogic $\mathcal{O} \subseteq \text{HML}$. Then, $\preceq_{\mathcal{O}}$ is a preorder, and $\sim_{\mathcal{O}}$ an equivalence relation.*³⁸

Lemma 2.4 (HML simulation). *Hennessy–Milner logic equivalence \sim_{HML} is a simulation relation.*³⁹

Proof. Assume it were not. Then there would need to be $p \sim_{\text{HML}} q$ with step $p \xrightarrow{\alpha} p'$, and no q' such that $q \xrightarrow{\alpha} q'$ and $p' \sim_{\text{HML}} q'$. So there would need to be a distinguishing formula $\varphi_{q'}$ for each q' that q can reach by an α -step. Consider the formula $\varphi_{\alpha} := \langle \alpha \rangle \bigwedge_{q' \in \text{Der}(q, \alpha)} \varphi_{q'}$. It must be true for p and false for q , contradicting $p \sim_{\text{HML}} q$. \square

³⁴  primrec [Hennessy_Milner_Logic.lts](#)
.satisfies

³⁵  abbreviation [LTS_Semantics.lts](#)
.distinguishes

³⁶  definition [LTS_Semantics.lts.preordered](#)

³⁷  definition [LTS_Semantics.lts.equivalent](#)

³⁸  lemma [LTS_Semantics.lts.equivalent](#)
_equiv

³⁹  lemma [HML_Spectrum.lts.hml_equiv](#)
_sim

⁴⁰  lemma [HML_Spectrum.lts.hml_bisim](#)
_invariant

Lemma 2.5 (Bisimulation invariance). *If $p \in \llbracket \varphi \rrbracket$ and $p \sim_B q$ then $q \in \llbracket \varphi \rrbracket$.*⁴⁰

Proof. Induct over the structure of φ with arbitrary p and q .

- Case $p \in \llbracket \langle \alpha \rangle \varphi \rrbracket$. Thus there is $p' \in \llbracket \varphi \rrbracket$ with $p \xrightarrow{\alpha} p'$. Because \sim_B is a simulation according to Lemma 2.1, this implies q' with $p' \sim_B q'$. The induction hypothesis makes $p' \in \llbracket \varphi \rrbracket$ entail $q' \in \llbracket \varphi \rrbracket$ and thus $q \in \llbracket \langle \alpha \rangle \varphi \rrbracket$.
- Case $p \in \llbracket \bigwedge_{i \in I} \varphi_i \rrbracket$. The induction hypothesis on the φ_i directly leads to $q \in \llbracket \bigwedge_{i \in I} \varphi_i \rrbracket$.
- Case $p \in \llbracket \neg \varphi \rrbracket$. Symmetry of \sim_B according to Proposition 2.2, implies $q \sim_B p$. By induction hypothesis, $q \in \llbracket \varphi \rrbracket$ implies $p \in \llbracket \varphi \rrbracket$. So, using contraposition, the case implies $q \in \llbracket \neg \varphi \rrbracket$. \square

Combining bisimulation invariance for one direction and that \sim_{HML} is a symmetric simulation (Proposition 2.5 and Lemma 2.4) for the other, we obtain that HML precisely characterizes bisimilarity:

Theorem 2.1 (Hennessy–Milner theorem). *Bisimilarity and HML equivalence coincide, that is, $p \sim_B q$ precisely if $p \sim_{\text{HML}} q$.*⁴¹

⁴¹  [theorem HML_Spectrum.lts.Hennessy_Milner_theorem](#)

Remark 2.3 (Infinitary conjunctions). In the standard presentation of the Hennessy–Milner theorem, image finiteness of the transition system is assumed. This means that $\text{Der}(p, \alpha)$ is finite for every $p \in \mathcal{P}$. The amount of outgoing transitions matters precisely in the construction of φ_α in the proof of Lemma 2.4. But as our definition of HML (Definition 2.11) allows infinite conjunctions $\bigwedge_{i \in I} \dots$, we do not need an assumption here. The implicit assumption is that the cardinality of index sets I can match that of $\text{Der}(p, \alpha)$. The original proof by Hennessy & Milner (1980) uses binary conjunction ($\varphi_1 \wedge \varphi_2$) and thus can only express finitary conjunctions.

2.3.3 The Perks of Modal Characterizations

There is merit in also characterizing other equivalences through sublogics $\mathcal{O} \subseteq \text{HML}$. Modal characterizations have four immediate big advantages:

Modal characterizations lead to *proper preorders and equivalences by design* due to Proposition 2.5. That is, if a behavioral preorder (or equivalence) is defined through modal logics, there is no need of proving transitivity and reflexivity (and symmetry).

Secondly, checking state equivalence for notions of HML sublogics *can soundly be reduced to checks on bisimulation-minimized systems* as the transformation does not affect the sets of observations for minimized states (by combining Theorem 2.1 and Proposition 2.4).

Thirdly, modal characterizations can directly *unveil the hierarchy between preorders*, if defined cleverly, because of the following property.

Proposition 2.6. *If $\mathcal{O} \subseteq \mathcal{O}'$ then $p \preceq_{\mathcal{O}'} q$ implies $p \preceq_{\mathcal{O}} q$ for all p, q .*⁴²

⁴²  [lemma LTS_Semantics.lts_semantics_preorder_contraposition](#)

Pay attention to the opposing directions of \subseteq and implication here!

Fourthly, as a corollary of Proposition 2.6, modal characterizations *ensure equivalences to be abstractions of bisimilarity*, which is a sensible base notion of equivalence.⁴³

In Chapter 3, we will discuss how the hierarchy of behavioral equivalences can be understood nicely and uniformly if viewed through the modal lens.

2.3.4 Expressiveness and Distinctiveness

Proposition 2.6 actually is a weak version of another proposition about *distinctiveness* of logics.

Definition 2.14 (Preorder of expressiveness). We say that an *Act*-observation language \mathcal{O} is *less or equal in expressiveness* to another \mathcal{O}' iff, for any *Act*-transition system, for each $\varphi \in \mathcal{O}$, there is $\varphi' \in \mathcal{O}'$ such that $\llbracket \varphi \rrbracket = \llbracket \varphi' \rrbracket$. (The definition can also be read with regards to a fixed transition system \mathcal{S} .)⁴⁴ If the inclusion holds in both directions, \mathcal{O} and \mathcal{O}' are *equally expressive*.

Definition 2.15 (Preorder of distinctiveness). We say that an *Act*-observation language \mathcal{O} is *less or equal in distinctiveness* to another \mathcal{O}' iff, for any *Act*-transition system and states p and q , for each $\varphi \in \mathcal{O}$ that distinguishes p from q , there is $\varphi' \in \mathcal{O}'$ that distinguishes p from q .⁴⁵ If the inclusion holds in both directions, \mathcal{O} and \mathcal{O}' are *equally distinctive*.

Lemma 2.6. *Subset relationship entails expressiveness, which entails distinctiveness.*

- If $\mathcal{O} \subseteq \mathcal{O}'$, then \mathcal{O} is less or equal in expressiveness to \mathcal{O}' .⁴⁶
- If \mathcal{O} is less or equal in expressiveness to \mathcal{O}' , then \mathcal{O} is less or equal in distinctiveness to \mathcal{O}' .⁴⁷

The other direction does not hold. For instance, $\text{HML} \not\subseteq \text{HML} \setminus \{\top\}$, but they are equally expressive as $\neg\neg\top$ can cover for the removed element. At the same time, $\{\top\}$ is more expressive than \emptyset , but equally distinctive.

The stronger version of Proposition 2.6 is thus:

Proposition 2.7. *If \mathcal{O} is less or equal in distinctiveness to \mathcal{O}' then $p \preceq_{\mathcal{O}'} q$ implies $p \preceq_{\mathcal{O}} q$ for all p, q .⁴⁸*


Remark 2.4. Through the lens of expressiveness, we can also justify why our convention to implicitly flatten conjunctions is sound: As this does not change a conjunction's truth value, HML subsets with and without flattening-convention are equally expressive and thus distinctive.


Often, an equivalence may be characterized by different sublogics. In particular, one may find smaller characterizations as in the following example for bisimilarity, which will be relevant in the upcoming game characterizations.


⁴³ Among other things, bisimilarity checking has a better time complexity than other notions as will be discussed in Section 3.3.3.

⁴⁴  definition LTS_Semantics.lts_semantics.leq_expressive

⁴⁵  definition LTS_Semantics.lts_semantics.leq_distinctive

⁴⁶  lemma LTS_Semantics.lts_semantics.subset_expressiveness

⁴⁷  lemma LTS_Semantics.lts_semantics.expressiveness_entails_distinctiveness


⁴⁸  lemma LTS_Semantics.lts_semantics.preorder_expressiveness_contraposition

Definition 2.16 (Bisimulation observations). Consider $\mathcal{O}_{[B]} \subseteq \text{HML}$ described by the following grammar:

$$\begin{array}{lcl} \varphi^{[B]} & ::= & \langle \alpha \rangle \bigwedge_{i \in I} \varphi_i^{[B]} \\ & | & \neg \varphi^{[B]} \end{array}$$

$\mathcal{O}_{[B]}$ is a proper subset of HML. For instance, it lacks the observation $\bigwedge \{ \langle a \rangle, \langle b \rangle \}$. Due to the subset relation, $\mathcal{O}_{[B]}$ must be less or equal in expressiveness to HML, but this inclusion too is strict as \top cannot be covered for. But both logics are equally distinctive!

Lemma 2.7. $\mathcal{O}_{[B]}$ and HML are equally distinctive.⁴⁹

⁴⁹  theorem [HML_Spectrum.lts.html_and_bisimulation_game_observations_equally_distinctive](#)

Proof. One direction is immediate from Lemma 2.6 as $\mathcal{O}_{[B]} \subseteq \text{HML}$.

For the other direction, we need to establish that, for each $\varphi \in \text{HML}$ distinguishing some p from some q , there is a $\varphi' \in \mathcal{O}_{[B]}$ distinguishing p from q . We induct on the structure of φ with arbitrary p and q .

- Case $\langle \alpha \rangle \varphi$ distinguishes p from q . Thus there is p' such that $p \xrightarrow{\alpha} p'$ and that φ distinguishes p' from every $q' \in \text{Der}(q, \alpha)$. By induction hypothesis, there must be $\varphi'_{q'} \in \mathcal{O}_{[B]}$ distinguishing p' from q' for each q' . Thus $\langle \alpha \rangle \bigwedge_{q' \in \text{Der}(q, \alpha)} \varphi'_{q'} \in \mathcal{O}_{[B]}$ distinguishes p from q .
- Case $\bigwedge_{i \in I} \varphi_i$ distinguishes p from q . Therefore some φ_i already distinguishes p from q . By induction hypothesis, there must be $\varphi'_i \in \mathcal{O}_{[B]}$ distinguishing p from q .
- Case $\neg \varphi$ distinguishes p from q . Thus φ distinguishes q from p . By induction hypothesis, there is $\varphi' \in \mathcal{O}_{[B]}$ distinguishing q from p . Accordingly, $\neg \varphi' \in \mathcal{O}_{[B]}$ distinguishes p from q . \square

The smaller bisimulation logic $\mathcal{O}_{[B]}$ will turn out to be closely related to the game characterization of bisimilarity in Section 2.4.5.

2.4 Games

So far, we have only seen behavioral equivalences and modal formulas as mathematical objects and not cared about decision procedures. This section introduces *game-theoretic characterizations* as a way of easily providing decision procedures for equivalences and logics alike. Intuitively, the games can be understood as dialogs between a party that tries to defend a claim and a party that tries to attack it.

2.4.1 Reachability Games

We use *Gale–Stewart-style reachability games* (in the tradition of [Gale & Stewart, 1953](#)) where the defender wins all infinite plays.

Definition 2.17 (Reachability game). A *reachability game* $\mathcal{G} = (G, G_d, \succrightarrow)$ is played on a directed graph consisting of

- a set of *game positions* G , partitioned into
 - *defender positions* $G_d \subseteq G$
 - and *attacker positions* $G_a := G \setminus G_d$,
- and a set of *game moves* $\succrightarrow \subseteq G \times G$.⁵⁰

Definition 2.18 (Plays and wins). We call the paths $g_0g_1\ldots \in G^\infty$ with $g_i \succrightarrow g_{i+1}$ *plays* of \mathcal{G} from g_0 , where G^∞ stands for finite and infinite words over G . The defender *wins* infinite plays. If a finite play $g_0 \ldots g_n \not\succrightarrow$ is stuck, the stuck player loses: The defender wins the play if $g_n \in G_a$, and the attacker wins if $g_n \in G_d$.

Usually, games are nondeterministic, that is, players have choices how a play proceeds at their positions. The player choices are formalized by *strategies*:

Definition 2.19 (Strategies and winning strategies). An *attacker strategy* is a (partial) function mapping play fragments ending at attacker positions to next positions to move to, $s_a: G^*G_a \rightarrow G$, where $g_a \succrightarrow s_a(\rho g_a)$ must hold for all ρg_a where s_a is defined.

A play $g_0g_1\ldots \in G^\infty$ is consistent with an attacker strategy s_a if, for all its prefixes $g_0\ldots g_i$ ending in $g_i \in G_a$, $g_{i+1} = s_a(g_0\ldots g_i)$.

Defender strategies are defined analogously, $s_d: G^*G_d \rightarrow G$.

If s ensures a player to win, s is called a *winning strategy* for this player. The player with a winning strategy from g_0 is said to *win* game \mathcal{G} from g_0 .

Usually, we will focus on positional strategies, that is, strategies that only depend on the most recent position, which we will type $s_a: G_a \rightarrow G$ (or $s_d: G_d \rightarrow G$, respectively).

We call the positions where a player has a winning strategy their winning region.

Definition 2.20 (Winning region). The set $\text{Win}_a \subseteq G$ of all positions g where the attacker wins in \mathcal{G} is called the *attacker winning region* of \mathcal{G} . The defender winning region Win_d is defined analogously.

Example 2.13 (A simple choice). Inspect the game in Figure 2.11, where round nodes represent defender positions and rectangular ones attacker positions. Its valid plays starting from (1) are (1), (1)[2a], (1)[2b], and (1)[2a](3). The defender can win from (1) with a strategy moving to [2b] where the attacker is stuck. Moving to [2a] instead would get the defender stuck. So, the defender winning region is $\text{Win}_d = \{(1), [2b]\}$ and the attacker wins $\text{Win}_a = \{[2a], (3)\}$. In Figure 2.11, the winning regions are marked with blue for the defender and red for the attacker.

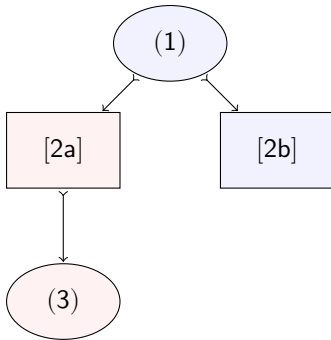


Figure 2.11: A simple game.

The games we consider are *positionally determined*. This means, for each possible initial position, exactly one of the two players has a positional winning strategy s .

Proposition 2.8 (Determinacy). *Reachability games are positionally determined, that is, for any game, each game position has exactly one winner: $G = \text{Win}_a \cup \text{Win}_d$ and $\text{Win}_a \cap \text{Win}_d = \emptyset$, and they can win using a positional strategy.*⁵¹

We care about reachability games because they are versatile in characterizing formal relations. Everyday inductive (and coinductive) definitions can easily be encoded as games as in the following example.

Example 2.14 (The \leq -Game). Consider the following recursive definition of a less-or-equal operator \leq on natural numbers in some functional programming language. (Consider nat to be defined as recursive data type $\text{nat} = 0 \mid \text{Succ nat}$.)

```
( 0 ≤ m )      = True
(Succ n ≤ 0 )   = False
(Succ n ≤ Succ m) = (n ≤ m)
```

We can think of this recursion as a game where the attacker tries to prove that some natural number on the left is bigger than the right by always decrementing the left number and challenging the defender to do the same for the right stack too. Whoever hits zero first, loses.

This means we distribute the roles such that the defender wins for output `True` and the attacker for output `False`. The two base cases need to be dead ends for one of the players.

Formally, the game \mathcal{G}_{leq} consists of attacker positions $[n, m]$ and defender positions (n, m) for all $n, m \in \mathbb{N}$, connected by chains of moves:

$$\begin{aligned} [n+1, m] &\rightarrow_{\text{leq}} (n, m) \\ (n, m+1) &\rightarrow_{\text{leq}} [n, m]. \end{aligned}$$

An excerpt of the game graph below $[3, 1]$ and $[1, 2]$ is shown in Figure 2.12.

\mathcal{G}_{leq} now characterizes \leq on \mathbb{N} in the sense that: The defender wins \mathcal{G}_{leq} from $[n, m]$ precisely if $n \leq m$. (Proof: Induct over n with arbitrary m .)

The game is *boring* because the players do not ever have any choices. They just count down their way through the natural numbers until they hit $[0, m-n]$ if $n \leq m$, or $(n-m-1, 0)$ otherwise.

\mathcal{G}_{leq} is quite archetypical for the preorder and equivalence games we will use in the following pages. But do not worry, the following games will demand the players to make choices.

2.4.2 The Semantic Game of HML

As first bigger example of how recursive definitions can be turned into games, let us quickly look at a standard way of characterizing the semantics of HML

⁵¹ This is just an instantiation of positional determinacy of parity games (Zielonka, 1998). Parity games extend reachability games by number-coloring of positions. The defender only wins the infinite plays where the least color that appears infinitely often is even. Reachability games are the subclass of parity games only colored by 0.

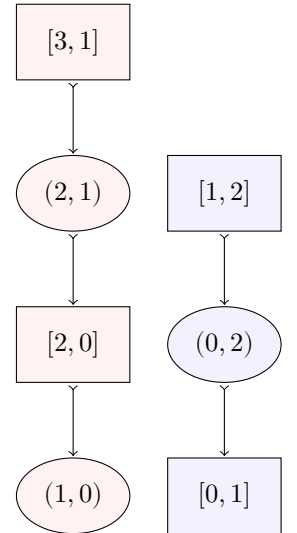


Figure 2.12: A part of \mathcal{G}_{leq} .

(Definition 2.12) through a game. The defender wins precisely if the game starts from a state and formula such that the state satisfies the formula.

Definition 2.21 (HML game). For a transition system $\mathcal{S} = (\mathcal{P}, \text{Act}, \rightarrow)$, the HML game $\mathcal{G}_{\text{HML}}^{\mathcal{S}} = (G_{\text{HML}}, G_d, \rightarrow_{\text{HML}})$ is played on $G_{\text{HML}} = \mathcal{P} \times \text{HML}$, where the defender controls observations and negated conjunctions, that is $(p, \langle \alpha \rangle \varphi) \in G_d$ and $(p, \neg \bigwedge_{i \in I} \varphi_i) \in G_d$ (for all φ, p, I), and the attacker controls the rest.

- The defender can perform the moves:

$$\begin{array}{lll} (p, \langle \alpha \rangle \varphi) & \rightarrow_{\text{HML}} & (p', \varphi) \quad \text{if } p \xrightarrow{\alpha} p' \quad \text{and} \\ (p, \neg \bigwedge_{i \in I} \varphi_i) & \rightarrow_{\text{HML}} & (p, \neg \varphi_i) \quad \text{with } i \in I; \end{array}$$

- and the attacker can move:

$$\begin{array}{lll} (p, \neg \langle \alpha \rangle \varphi) & \rightarrow_{\text{HML}} & (p', \neg \varphi) \quad \text{if } p \xrightarrow{\alpha} p' \quad \text{and} \\ (p, \bigwedge_{i \in I} \varphi_i) & \rightarrow_{\text{HML}} & (p, \varphi_i) \quad \text{with } i \in I \quad \text{and} \\ (p, \neg \neg \varphi) & \rightarrow_{\text{HML}} & (p, \varphi). \end{array}$$

The intuition is that *disjunctive* constructs ($\langle \cdot \rangle \dots, \neg \bigwedge \dots$) make it easier for a formula to be true and thus are controlled by the defender who may choose which of the ways to show truth is most convenient. At *conjunctive* constructs ($\neg \langle \cdot \rangle \dots, \bigwedge \dots$) the attacker chooses the option that is the easiest to disprove. The most trivial case for the attacker to lose is (p, \top) .

Example 2.15. The game of Example 2.13 is exactly the HML game $\mathcal{G}_{\text{HML}}^{\mathcal{S}_{\text{PQ}}}$ for formula $\langle \tau \rangle \neg \langle a \rangle \top$ and state P of Example 2.12 with (1) $:= (P, \langle \tau \rangle \neg \langle a \rangle \top)$, [2a] $:= (p_a, \neg \langle a \rangle \top)$, [2b] $:= (p_b, \neg \langle a \rangle \top)$, and (3) $:= (p_1, \neg \top)$.

The defender winning region $\text{Win}_d = \{(P, \langle \tau \rangle \neg \langle a \rangle \top), (p_b, \neg \langle a \rangle \top)\}$ corresponds to the facts that $P \in \llbracket \langle \tau \rangle \neg \langle a \rangle \top \rrbracket$ and $p_b \in \llbracket \neg \langle a \rangle \top \rrbracket$.

As the technicalities are tangential to this thesis, we state the characterization result without proof:⁵²

Lemma 2.8. *The defender wins $\mathcal{G}_{\text{HML}}^{\mathcal{S}}$ from (p, φ) precisely if $p \in \llbracket \varphi \rrbracket$.*

2.4.3 The Bisimulation Game

We now can add the bottom layer of Figure 2.1: That bisimilarity can be characterized through a game, where the defender wins if the game starts on a pair of bisimilar states. This approach has been popularized by Stirling (1996).

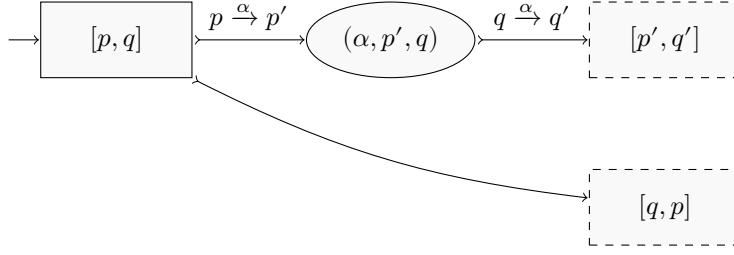
Definition 2.22 (Bisimulation game). For a transition system \mathcal{S} , the *bisimulation game* $\mathcal{G}_{\text{B}}^{\mathcal{S}}$ is played on attack positions $G_a := \mathcal{P} \times \mathcal{P}$ and defense positions $G_d := \text{Act} \times \mathcal{P} \times \mathcal{P}$ with the following moves:⁵³

- The attacker may *challenge simulation*

$$[p, q] \rightarrow_{\text{B}} (\alpha, p', q) \quad \text{if } p \xrightarrow{\alpha} p',$$

⁵² A detailed presentation of a more general HML game, also extending to recursive HML, can be found in Wortmann et al. (2015, Chapter 3).

⁵³ locale [Equivalence_Games.bisim_game](#)

Figure 2.13: Game scheme of the bisimulation game \mathcal{G}_B (Definition 2.22).

- or the attacker may *swap sides*

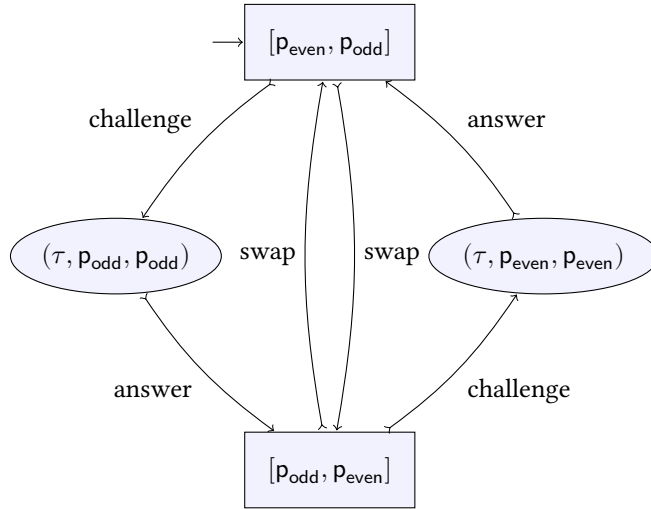
$$[p, q] \rightsquigarrow_B [q, p],$$

- and the defender *answers simulation challenges*

$$(\alpha, p', q) \rightsquigarrow_B [p', q'] \quad \text{if } q \xrightarrow{\alpha} q'.$$

A schematic depiction of the game rules can be seen in Figure 2.13. From dashed nodes, the game proceeds analogously to the initial attacker position.

Example 2.16. Consider $p_{\text{even}} \sim_B p_{\text{odd}}$ of Example 2.8. The bisimulation game on this system is given by Figure 2.14:

Figure 2.14: Bisimulation game under $p_{\text{even}}, p_{\text{odd}}$. Moves are annotated with the game rules from which they derive.

Clearly, there is no way for the attacker to get the defender stuck. Whatever strategy the attacker chooses, the game will go on forever, leading to a win for the defender. That it is always safe for the defender to *answer* with moves to $[p_{\text{even}}, p_{\text{odd}}]$ and $[p_{\text{odd}}, p_{\text{even}}]$ corresponds to $\mathcal{R} := \{(p_{\text{even}}, p_{\text{odd}}), (p_{\text{odd}}, p_{\text{even}})\}$

Remark 2.5 (Playing equivalence games). One of the big advantages of game characterizations is that they provide a way to discuss equivalence and inequivalence interactively among humans. There also are several computer game implementations of bisimulation games.

For instance, Peacock (2020) implements a game about simulation and bisimulation as well as several weaker notions. The human plays the attacker trying to point out the inequivalence of systems according to the rules of Definition 2.22. Figure 2.16 shows a screenshot of said game. It can be played on <https://www.concurrency-theory.org/rvg-game/>.

Remark 2.6 (Number comparison as simulation game). If one plays the bisimulation game of Definition 2.22 without the swapping moves, it will characterize the simulation preorder.

Consider the family of processes N_n with $n \in \mathbb{N}$. Define $N_0 := \mathbf{0}$ and $N_{n+1} := \text{one}.N_n$. Then, the simulation game played from $[N_n, N_m]$ is isomorphic to the \leq -game \mathcal{G}_{leq} from Example 2.14. Therefore, the defender wins $[N_n, N_m]$ in the simulation game precisely if they win $[n, m]$ in the \leq -game. We can compare numbers $n \leq m$ by comparing programs $N_n \preceq_S N_m$!

In this sense, comparisons of programs and of numbers are ... comparable.

2.4.4 Deciding Reachability Games

All we need to turn a game characterization into a decision procedure is a way to decide which player wins a position. With this, \mathcal{G}_{HML} of Definition 2.21 entails a model checking procedure for HML and \mathcal{G}_{B} a bisimulation checking algorithm on finite-state systems.

Algorithm 2.1 describes how to compute who wins a finite reachability game for each position in time linear to the size of the game graph $O(|\rightarrow|)$.

Intuitively, `compute_winning_region` first assumes that the defender were to win everywhere and that each outgoing move of every position might be a winning option for the defender. Over time, every position that is determined to be lost by the defender is added to a `todo` list.⁵⁷

At first, the defender loses immediately exactly at the defender's dead ends. Each defender-lost position is added to the `attacker_win` set. To trigger the recursion, each predecessor is noted as defender-lost, if it is controlled by the attacker, or the amount of outgoing defender options is decremented if the predecessor is defender-controlled. If the count of a predecessor position hits 0, the defender also loses from there.

Once we run out of `todo` positions, we know that the attacker has winning strategies exactly for each position we have visited.

The following Table 2.1 lists how Algorithm 2.1 computes the winning region $\text{Win}_a = \{[2a], (3)\}$ of the simple choice game of Example 2.13.

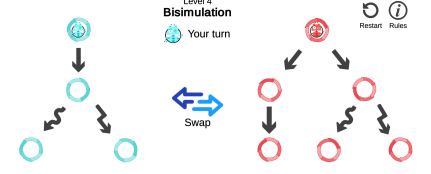


Figure 2.16: Screenshot of Peacock's bisimulation computer game. (In the computer game, the colors are coded inversely to our presentation. The blue character represents the attacker-controlled state.)

⁵⁷ Variants of this algorithm and explanation have also been used in Bisping (2018) and Bisping et al. (2022).

```

1  def compute_winning_region( $\mathcal{G} = (G, G_d, \succrightarrow)$ ):
2    defender_options :=  $[g \mapsto n \mid g \in G_d \wedge n = |\{g' \mid g \succrightarrow g'\}|]$ 
3    attacker_win :=  $\emptyset$ 
4    todo :=  $\{g \in G_d \mid \text{defender\_options}[g] = 0\}$ 
5    while todo  $\neq \emptyset$ :
6      g := some todo
7      todo := todo  $\setminus \{g\}$ 
8      if g  $\notin$  attacker_win:
9        attacker_win := attacker_win  $\cup \{g\}$ 
10     for  $g_p \in (\cdot \succrightarrow g)$ :
11       if  $g_p \in G_a$ :
12         todo := todo  $\cup \{g_p\}$ 
13       else :
14         defender_options[ $g_p$ ] := defender_options[ $g_p$ ] - 1
15         if defender_options[ $g_p$ ] = 0:
16           todo := todo  $\cup \{g_p\}$ 
17   Wina := attacker_win
18   return Wina

```

Algorithm 2.1: Deciding the attacker winning region Win_a of a reachability game \mathcal{G} in linear time of $|\succrightarrow|$ and linear space of $|G|$.

Table 2.1: Solving the game of Example 2.13.

g	defender_options	todo
-	(1) \mapsto 2, (3) \mapsto 0	(3)
(3)	(1) \mapsto 2, (3) \mapsto 0	[2a]
[2a]	(1) \mapsto 1, (3) \mapsto 0	\emptyset

As this game corresponds to the HML game by Example 2.15, the computation that the defender wins (1) checks that $\langle \tau \rangle \neg \langle a \rangle \top$ is true for state P of Example 2.12.

The inner loop of Algorithm 2.1 clearly can run at most $|\succrightarrow|$ -many times. Using sufficiently clever data structures, the algorithm hence shows:

Proposition 2.9. *Given a finite reachability game $\mathcal{G} = (G, G_d, \succrightarrow)$, the attacker's winning region $\text{Win}_a^{\mathcal{G}}$ (and $\text{Win}_d^{\mathcal{G}}$ as well) can be computed in $O(|\succrightarrow|)$ time and $O(|G|)$ space.*

Everything we can characterize as a reachability game, can thus be easily decided.

So, by now we know a world where equivalences are nice to characterize, *modal logics*, and a world where they are nice to compute: *games*. Obviously, there must be a link! Making this link explicit yields the *core proof approach* for the rest of this thesis.

2.4.5 How Bisimulation Game and HML Are Connected

Bisimulation game and Hennessy–Milner logic connect in a beautiful way. This connection usually receives less attention than Hennessy–Milner theorem and Stirling’s characterization. But it is the key insight for the main results of later chapters.

Let us briefly imagine a world without a dedicated bisimulation game. The Hennessy–Milner theorem implies that we could directly use the HML game of Definition 2.21 to describe bisimilarity:

Definition 2.23 (Naive bisimulation game). We extend the HML game of Definition 2.21 by the following prefix:

1. To challenge $[p, q]$, the attacker picks a formula $\varphi \in \text{HML}$ (claiming that φ distinguishes the states) and yields to the defender (φ, p, q) .
2. The defender decides where to start the HML game:
 1. Either at $(p, \neg\varphi)$ (claiming φ to be non-distinguishing because it is untrue for p)
 2. or at (q, φ) (claiming φ to be non-distinguishing because it is true for q).
3. After that, the turns proceed as prescribed by Definition 2.21.

This naive game, too, has the property that the defender wins from $[p, q]$ iff $p \sim_B q$. The downside of the game is that the attacker has infinitely many options $\varphi \in \text{HML}$ to pick from!

The proper bisimulation game of Definition 2.22, on the other hand, is *finite for finite transition systems*. Therefore, it induces decision procedures by the reasoning of Section 2.4.4.

We will now argue that the *bisimulation game actually is a variant of the naive game*, where (1) the attacker names their formula *gradually*, and (2) the formulas stem from $\mathcal{O}_{[B]} \subseteq \text{HML}$ of Definition 2.16. To this end, we will show that attacker’s winning strategies imply distinguishing formulas, and that a distinguishing formula from $\mathcal{O}_{[B]}$ certifies the existence of winning attacker strategies.

Example 2.18 (Formulating attacks). Let us illustrate how to derive distinguishing formulas using the game of Example 2.17.

Recall that the attacker wins by moving $[Q, T] \succrightarrow_B [T, Q] \succrightarrow_B (\tau, \textcircled{a}, Q) \succrightarrow_B [\textcircled{a}, q_{ab}] \succrightarrow_B [q_{ab}, \textcircled{a}] \succrightarrow_B (a, q_1, \textcircled{a}) \not\succrightarrow_B$. In Figure 2.17, we label the game nodes with the (sub-)formulas this strategy corresponds to. The cloud indicates where we omit some region where the defender wins. Swap moves become negations, and simulation moves become observations with a conjunction of formulas for each defender option. This attacker strategy can thus be expressed by $\neg\langle\tau\rangle \wedge \{\neg\langle a \rangle \top\} \in \mathcal{O}_{[B]}$. We will call such formulas “strategy formulas.”

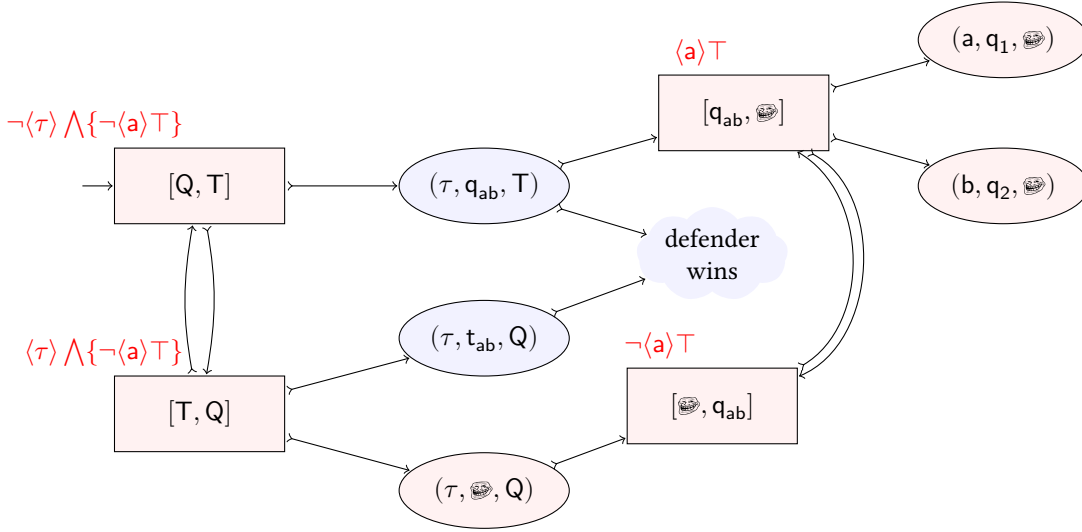


Figure 2.17: The bisimulation game of Example 2.18 with attacker formulas.

More generally, the following lemmas explain the construction of distinguishing formulas from attacker strategies, and back. They are a blueprint of game characterization proofs throughout the thesis.

We will refer to the property that attacker wins in a game represent distinguishing formulas as “distinction soundness” of a game.

Lemma 2.9 (Distinction soundness). *Let function s be a positional winning strategy for the attacker on \mathcal{G}_B from $[p, q]$. Construct formulas recursively from game positions, $\varphi_s(g)$, as follows:*

$$\varphi_s([p, q]) := \begin{cases} \neg\varphi_s([q, p]) & \text{if } s([p, q]) = [q, p] \\ \langle\alpha\rangle \wedge \{\varphi_s([p', q']) \mid q \xrightarrow{\alpha} q'\} & \text{if } s([p, q]) = (\alpha, p', q) \end{cases}$$

Then φ_s is defined for $[p, q]$ and distinguishes p from q . Also, $\varphi_s([p, q]) \in \mathcal{O}_{[B]}$.

Proof.

1. The recursive construction works to define $\varphi_s([p, q])$ as s must induce a well-founded order on game positions⁵⁸ in order for the attacker to win, and as the recursive invocations remain in the attacker winning strategy.
2. The distinction can be derived by induction on the construction of $\varphi_s([p, q])$: Assume for p', q' appearing in recursive invocations of the definition of φ_s that $\varphi_s([p', q'])$ distinguishes them and that $\varphi_s([p', q']) \in \mathcal{O}_{[B]}$. We have to show that the constructions in φ_s still fulfill this property:

⁵⁸ An order on a set A is well-founded iff each nonempty subset $A' \subseteq A$ has a minimal element $\min A'$. In the game, the minima correspond to the defender dead ends the attacker is navigating towards.

- As $\varphi_s([p', q'])$ distinguishes p' from q' , its negation $\neg\varphi_s([p', q'])$ must distinguish q' from p' by the HML semantics. Moreover, the grammar of Definition 2.21 is closed under negation.
- For $\langle\alpha\rangle \wedge \{\varphi_s([p', q']) \mid q \xrightarrow{\alpha} q'\}$ to distinguish p from q , there must be $p \xrightarrow{\alpha} p^* \in \llbracket \wedge \{\varphi_s([p', q']) \mid q \xrightarrow{\alpha} q'\} \rrbracket$, and for all q^* with $q \xrightarrow{\alpha} q^*$ it must be that $q^* \notin \llbracket \wedge \{\varphi_s([p', q']) \mid q \xrightarrow{\alpha} q'\} \rrbracket$. We choose $p^* = p'$, because, as s is an attacker winning strategy, $s([p, q]) = (\alpha, p', q)$ is a valid move with $p \xrightarrow{\alpha} p'$. Also, all q' the defender may select in answers remain in the winning domain of the attacker, and we may use the induction hypothesis that $q' \notin \llbracket \varphi_s([p', q']) \rrbracket$. Therefore, for each q^* there is a false conjunct with $q' = q^*$ in $\wedge \{\varphi_s([p', q']) \mid q \xrightarrow{\alpha} q'\}$ that proves $q^* \notin \llbracket \wedge \{\varphi_s([p', q']) \mid q \xrightarrow{\alpha} q'\} \rrbracket$. \square

We say that a game moreover is distinction-complete if distinguishing formulas of a sublogic can be mirrored by attacker strategies.

Lemma 2.10 (Distinction completeness). *If $\varphi \in \mathcal{O}_{[B]}$ distinguishes p from q , then the attacker wins from $[p, q]$.*

Proof. By induction on the derivation of $\varphi \in \mathcal{O}_{[B]}$ according to the definition from Definition 2.16 with arbitrary p and q .

- Case $\varphi = \langle\alpha\rangle \bigwedge_{i \in I} \varphi_i$. As φ distinguishes p from q , there must be a $p' \xrightarrow{\alpha} p'$ such that $\bigwedge_{i \in I} \varphi_i$ distinguishes p' from every $q' \in \text{Der}(q, \alpha)$. That is, for each $q' \in \text{Der}(q, \alpha)$, at least one $\varphi_i \in \mathcal{O}_{[B]}$ must be false. By induction hypothesis, the attacker thus wins each $[p', q']$. As these attacker positions encompass all successors of (α, p', q) , the attacker also wins this defender position and can win from $[p, q]$ by moving there with a simulation challenge.
- Case $\varphi = \neg\varphi'$. As φ distinguishes p from q , φ' distinguishes q from p . By induction hypothesis, the attacker wins $[q, p]$. So they can also win $[p, q]$ by performing a swap. \square

Lemma 2.9 and Lemma 2.10, together with the fact that $\mathcal{O}_{[B]}$ and HML are equally distinctive (Lemma 2.7), yield:

Theorem 2.3. *The attacker wins \mathcal{G}_B from $[p, q]$ precisely if there is a formula $\varphi \in \text{HML}$ distinguishing p from q .*

With Theorem 2.3, we have added the last arrow on the right side of Figure 2.1.

Of course, Theorem 2.3 could also be arrived at by gluing together the Hennessy–Milner theorem on bisimulation (Theorem 2.1) and Stirling’s bisimulation game characterization (Theorem 2.2), modulo the determinacy of games. But Theorem 2.3 transports a deeper insight:

The bisimulation game actually is a game of an attacker assembling a distinguishing formula from $\mathcal{O}_{[B]}$. The game rules for attacker moves follow the

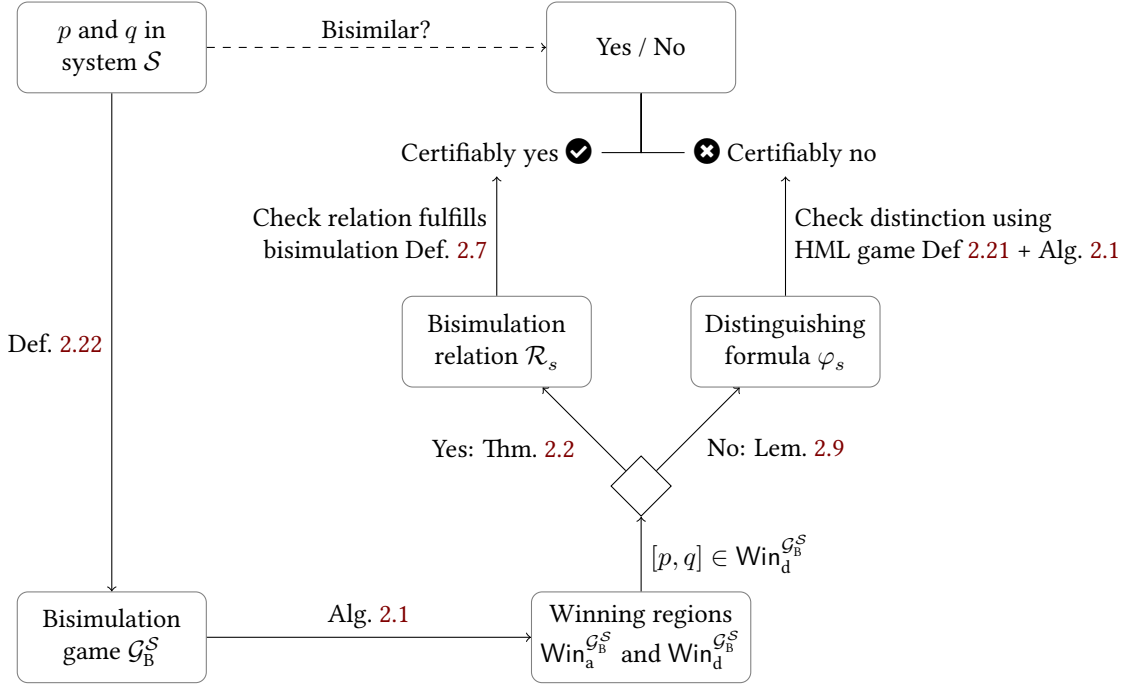


Figure 2.18: Checking bisimilarity and providing certificates.

grammar of $\mathcal{O}_{[B]}$. But parts of the HML semantics are baked into the game: Intuitively, the attacker must propose formulas that are true-by-construction for the left-hand state. To disprove the distinction, the defender may point out how the formula is also satisfied by the right-hand state. Negation turns the tables.

The rest of this thesis is about leveraging this intuition to address every behavioral equivalence. We will meet Theorem 2.3 again three times—each time in a more abstract form.⁵⁹

2.5 Discussion

This chapter has taken a tour of characterizations for standard notions of behavioral preorder and equivalence on transition systems such as trace equivalence and bisimilarity. The aim of this has been to prepare a generic framework to characterize and decide behavioral equivalences. So far, we have only instantiated the framework for bisimilarity, but the rest will follow.

Perspicuous algorithmics. In constructing the relationships of Figure 2.1 for bisimilarity, we have collected the theoretical ingredients for a *certifying algorithm* to check bisimilarity of states.

Figure 2.18 describes how to not only answer the question whether two states p and q are bisimilar, but how to also either provide a bisimulation

⁵⁹ And for the most abstract incarnation, there will even be an Isabelle/HOL formalization in Section 7.1.3.

relation or a distinguishing formula for the two as certificates for the answer. (The arrows stand for computations, the boxes for data.)

In this view, the game of possible distinctions and their preventability between attacker and defender serves as the “ground truth” about the bisimilarity of two states. Bisimulation relations and modal formulas only appear as certificates of defender and attacker strategies, mediating between game and transition system. The Hennessy–Milner theorem emerges on this level as a *shadow of the determinacy* of the bisimulation game (Idea 2). This whole framework draws heavily from Stirling (1996). The following chapters will reveal how the game framework generalizes to check multiple equivalences.

Alternatives. There are alternative frameworks for deciding behavioral equivalences that this thesis will not be using:

- **Fixed-point iteration.** Simulation-like behavioral equivalences and preorders can be expressed as greatest fixed points of monotonic functions (Aceto et al., 2007, Section 4.3). Fixed-point iteration computes such relations on finite-state transition systems by initially assuming all states are related and iteratively removing pairs that violate the (bi-)simulation condition. This process continues until no such pairs remain. In effect, a chain reaction happens analogous to the one that Algorithm 2.1 performs on the bisimulation game. Compared to the game approach, fixed-point iteration usually is slower as it lacks knowledge about logical connections between pairs in the candidate relation.
- **Partition refinement.** The fastest algorithms for bisimilarity use partition refinement (Paige & Tarjan, 1987). Partitions represent equivalence relations as colorings on graphs, which is linear in the state space $|\mathcal{P}|$ (opposed to quadratic combinations for pairs in relations for fixed points and attacker positions for games). Intuitively, parts of the partitions are split up repeatedly, leading to a similar iterative refinement as in the general fixed-point approach. The splits are also informed by modal distinctions as explicated by Cleaveland (1991). But clever decisions of where to split next and the leaner representation of data allow for more efficient algorithmics. However, the nice approach only works if the fixed-point characterization ensures intermediate relations to be *symmetric*. This is only the case for bisimilarity, thus heavily restricting the direct applicability of partition refinement.
- **Reduction to bisimilarity.** Other equivalences are usually checked by transforming the transition system in some way to make bisimilarity and the respective equivalence coincide on the system (Cleaveland & Sokolsky, 2001, Section 3.4). A standard example would be to make the transition system deterministic via subset construction, after which trace equivalence can be checked as bisimilarity (Cleaveland & Hennessy, 1993).

- **Term rewriting.** If working on process-algebraic terms and similar formalisms, equivalence can also be established through equational reasoning (Mayr, 2000). This approach requires a behavioral *congruence* with an axiomatic characterization which processes to consider as equal. Intuitively, the algorithm then searches ways of rewriting one process into another. But the rules usually create an infinite search space, only allowing semi-decision procedures. Together with the coupling to specific process calculi, the approach thus is better fit for the context of proof assistants than for general equivalence checkers. However, proofs on axiomatic characterizations often contain normalizations that can inform the design of reductions to bisimilarity.
- **Characteristic formulas.** Equivalence can also be handled through characteristic modal formulas (Steffen, 1989). Intuitively, one constructs a formula for a state that is precisely true for all conceivable equivalent states. Checking equivalence with another state then boils down to model checking the formula on that state. However, this approach requires a more complex language of *recursive* modal formulas and remains mostly academic, as no major tools use it for equivalence checking.

Modal logics point the way. We have observed that behavioral equivalences form hierarchies and that these hierarchies are handled nicely using modal characterizations to rank distinguishing powers (Section 2.3.3).

We have also seen how to link game rules to productions in a language of potentially distinguishing formulas (Section 2.4.5). This departs from common textbook presentations that motivate the bisimulation game through the relational (coinductive) characterization (e.g. Sangiorgi, 2012). In later chapters, we will rather derive equivalence games from grammars of modal formulas (Idea 3), to generalize the framework of Figure 2.18 from bisimilarity to whole spectra of equivalence.

But first, we have to make formal what we mean by *spectra of behavioral equivalence*.

3 Context: The Spectrum of Equivalences

We will now take a deeper dive into the question how groups of behavioral equivalences and preorders can be ranked in equivalence spectra.

For the following chapters, we will focus on the main “linear-time–branching-time spectrum” for the semantics of concrete processes treated by van Glabbeek (1990), the so-called “strong spectrum.” However, we will order the equivalences using the approach of parameterized *notions of observability* from the later “weak spectrum” by van Glabbeek (1993).

Section 3.1 provides the background on observability hierarchies, which motivate the chapter’s first core idea:

i Idea 4: Notions of observability bring order

Groups of equivalences can be defined and ranked in lattices of notions of observability.

In particular, Section 3.2 will introduce such a spectrum, forming a hierarchy of modal logics for the *strong spectrum*. From there, we quickly run into our core question:

i Idea 5: We have a spectroscopy problem

We can ask what notions of equivalence from a spectrum are the most fitting to relate two states.

Section 3.3 will define the *spectroscopy problem* formally and give lower bounds for its complexity on the strong spectrum.

3.1 Observability Hierarchies

Let us begin to pick up the idea from Section 2.3.3 that modal logics can nicely rank equivalences. The intuition is that HML sublogics capture what we consider to be *observable*. The more we mark as observable, the finer the resulting equivalence relations become.

Related publications. This chapter presents the spectrum of modal languages from “Process equivalence problems as energy games” (Bisping, 2023b) in a slightly clarified manner. The used hierarchy itself derives from Bisping et al. (2022). The alignment to the authoritative work by van Glabbeek (1990) has been formalized by Mattes (2024).

3.1.1 Understanding the Equivalence Hierarchy through Modal Logics

As promised, we revisit the hierarchy between bisimilarity, similarity, and trace equivalence of Section 2.2.3, modally. So far, we have only looked into the characterization of bisimilarity through the whole of HML in Theorem 2.1 or through $\mathcal{O}_{[B]}$ in Lemma 2.7. From now on, we will be working with *hierarchies of Hennessy–Milner theorems*.

Definition 3.1 (Logics of simulation and traces). We define the two HML sublogics \mathcal{O}_T , the *linear positive fragment**, and \mathcal{O}_S , the *positive fragment**, by the grammars starting at φ^T and φ^S .

$$\begin{aligned} \varphi^T &::= \langle \alpha \rangle \varphi^T \mid \top \\ \varphi^S &::= \langle \alpha \rangle \varphi^S \mid \bigwedge_{i \in I} \langle \alpha_i \rangle \varphi_i^S \end{aligned}$$

The logics characterize trace and simulation preorder (and equivalence):

Lemma 3.1 (Trace characterizations). $p \preceq_T q$ precisely if $p \preceq_{\mathcal{O}_T} q$.⁶⁰

Lemma 3.2 (Simulation characterization). $p \preceq_S q$ precisely if $p \preceq_{\mathcal{O}_S} q$.⁶¹

Clearly, $\mathcal{O}_T \subset \mathcal{O}_S \subset \text{HML}$. So, Proposition 2.6 that sublogics will equate at least as much as their parent logic, yields another way of establishing the entailment hierarchy between bisimilarity, similarity, and trace equivalence of Section 2.2.3.⁶²

In the modal realm, equivalences become naturally comparable. Contrast this with preceding definitions that live in different worlds: relational definition for (bi-)similarity of Definition 2.7 and denotational definition for trace equivalence of Definition 2.5. Heterogenous definitions lead to complicated proofs.


The modal view also reveals an intuitive hierarchy of “testing scenarios” for the equivalences, framed as *black box tests* in van Glabbeek (1990):


Trace equivalence matches an observer that can see *sequences of events*. They *just watch repeated executions* of the program, but are oblivious to possibilities and decisions.

Similarity matters to an experimenter that can also explore different *branches of possibilities*. This is valid if the experimenter can somehow *copy the system state* during the execution.

Bisimilarity captures that the experimenter can moreover determine if a future course of events is *impossible at some point*. This means that the experimenter can not only copy the execution state but also exhaustively test *every possibility* of how the system may continue.

But such levels of observability do not need to be linear, as we will see in the next subsection ...

⁶⁰  theorem [HML_Spectrum.lts.observations_traces_characterizes_trace_preorder](#)

⁶¹  theorem [HML_Spectrum.lts.observations_simulation_characterize_simulation_preorder](#)

⁶² Taking the two preceding lemmas together with Theorem 2.1 for \sim_B and HML.

3.1.2 Incomparabilities

Often, things we compare are comparable with respect to *different dimensions*. For instance, one can compare smartphone sizes with respect to width and height, as seen in Figure 3.1. This phenomenon of *orthogonal dimensions* also occurs for behavioral equivalences.

A well-known and natural notion of equivalence is that of *failure equivalence*. Intuitively, a failure says that the experimenter may follow a trace and see which actions are *impossible* at its end. Its standard definition is based on *failure* denotations:

Definition 3.2 (Failures). The set of failures of a process $\text{Failures}(p) \subseteq \text{Act}^* \times 2^{\text{Act}}$ is recursively defined as

- $((), X) \in \text{Failures}(p)$ if $X \cap \text{Ini}(p) = \emptyset$,
- $(\alpha\vec{w}, X) \in \text{Failures}(p)$ if there is p' with $p \xrightarrow{\alpha} p'$ and $(\vec{w}, X) \in \text{Failures}(p')$.

For instance, the failure $(\tau, \{a\})$ is in $\text{Failures}(P)$ but not in $\text{Failures}(Q)$ on Figure 2.3.

But would it not be nice if we could prevent the invention of new mathematical objects as denotations for each new notions of observability we consider? Fortunately, we can save the work, by directly employing modal logics:

Definition 3.3 (Failure logic). We define failure observations \mathcal{O}_F by the grammar:

$$\varphi^F ::= \langle \alpha \rangle \varphi^F \quad | \quad \bigwedge_{i \in I} \neg \langle \alpha_i \rangle \top$$

Clearly, this encompasses what we may observe via traces, but is something that we cannot observe using simulation observations. We consider \preceq_F given by $\preceq_{\mathcal{O}_F}$.

The distinguishing failure $(\tau, \{a\})$ can be expressed as $\langle \tau \rangle \bigwedge \{ \neg \langle a \rangle \top \} \in \mathcal{O}_F$ in HML. The formula distinguishes P from Q on Figure 2.3, $P \not\preceq_F Q$. This sets failures apart from simulation, as $P \preceq_S Q$ (cf. Example 2.6). On the other hand, no failure from \mathcal{O}_F distinguishes Q from P , implying $Q \preceq_F P$, but $Q \not\preceq_S P$.

As a consequence, simulation preorder and failure preorder are *incomparable*, that is, neither one implies the other. The same is true of the corresponding equivalences: similarity and failure equivalence. The situation is summed up by the non-linear hierarchy in Figure 3.2. After a quick glance at the diamond-like figure, it probably comes as no surprise what kind of mathematical structure we employ to handle the hierarchy: lattices.

3.1.3 Lattices

To handle non-linearity, we will work with *lattices* of notions of behavioral equivalence. The following definition gives the preliminaries to talk about this kind of partial orders.



Figure 3.1: Participants of ETAPS'22 discussing the spectrum of smartphone sizes. (Which is dominated by Hermanns's phone, at the bottom of the picture.) (Photo: ETAPS Association)

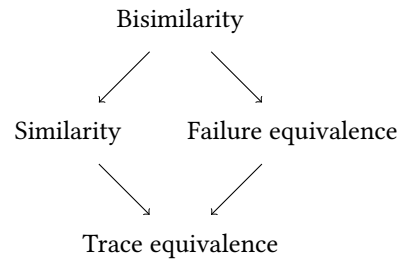


Figure 3.2: Equivalence hierarchy including failure equivalence.

Definition 3.4 (Lattices, bounds, chains). A *lattice* is a partially ordered set (B, \leq) , where there are greatest lower bounds $\inf\{b_1, b_2\}$ and least upper bounds $\sup\{b_1, b_2\}$ between each pair of elements $b_1, b_2 \in B$.

- The *greatest lower bound* of a set $B' \subseteq B$ is called its *infimum*, $\inf B'$. It refers to the greatest element $b \in B$ such that $b \leq b'$ for all $b' \in B'$.⁶³
- Dually, the *least upper bound* of a set $B' \subseteq B$ is called its *supremum*, $\sup B'$. It refers to the least element $b \in B$ such that $b \geq b'$ for all $b' \in B'$.
- For the pair-wise infimum we also use infix notation $b_1 \sqcap b_2 = \inf\{b_1, b_2\}$, and analogously $b_1 \sqcup b_2 = \sup\{b_1, b_2\}$.
- If a lattice (B, \leq) not only allows infima and suprema for pairs but for any set $B' \subseteq B$, it is called *complete*. We say inf-complete or sup-complete if only one of the two is true.
- We call a totally ordered subset $B' \subseteq B$ a *chain*.
- Dually, some $B' \subseteq B$ with no two elements $b_1, b_2 \in B'$ such that $b_1 \leq b_2$ is called an *anti-chain*.
- The *upward closure* $\uparrow B'$ of a $B' \subseteq B$ is defined as $\{b \in B \mid \exists b' \in B'. b' \leq b\}$. Analogously, the *downward closure* is given by $\downarrow B' := \{b \in B \mid \exists b' \in B'. b \leq b'\}$.

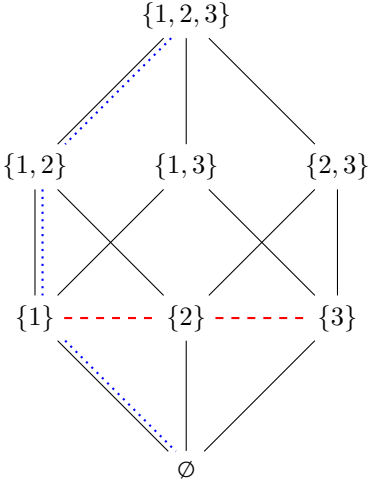


Figure 3.3: Lattice of subsets from Example 3.1. Solid lines stand for inclusion from bottom to top, transitive lines are left out.

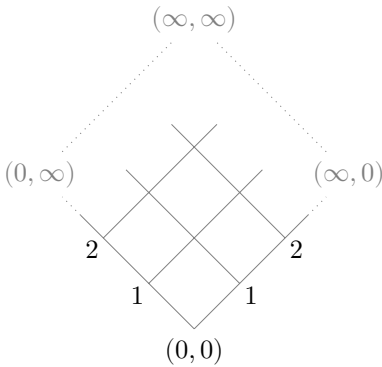


Figure 3.4: Visualization of the infinitary grid of \mathbb{N}^2 (and in gray, \mathbb{N}_∞^2) in Example 3.2.

Example 3.1 (Subset lattice). Given any set B , its subsets ordered by set inclusion $(2^B, \subseteq)$ form a complete lattice.

The *greatest lower bound* is given by set intersection $\bigcap B'$ with $B' \subseteq B$. The *least upper bound* is set union $\bigcup B'$ with $B' \subseteq B$. The empty set $\emptyset \in 2^B$ is the least element, $B \in 2^B$ is itself the greatest element.

Consider the subset lattice over $B = \{1, 2, 3\}$. It is “cube-like,” as can be seen in its Hasse diagram in Figure 3.3. An example of a (maximal) chain would be $\{\emptyset, \{1\}, \{1, 2\}, B\}$ (the nodes connected by a blue dotted line in the figure), because its members are ordered linearly $\emptyset \subset \{1\} \subset \{1, 2\} \subset B$. The set $\{\{1\}, \{2\}, \{3\}\}$ forms a (maximal) anti-chain (the nodes connected by a red dashed line in the figure), because its members do not include each other. Their respective subsets are chains / anti-chains as well.

Example 3.2 (Vector lattice). Given a linearly ordered set (B, \leq_B) , its d -ary Cartesian product with pointwise order (B^d, \leq) forms a lattice, where $b \leq b'$ iff $b_k \leq b'_k$ for all $k \in \{1, \dots, d\}$. Greatest lower bounds and least upper bounds can be transferred pointwise from B .

For instance, pairs of natural numbers, (\mathbb{N}^2, \leq) , form a lattice, as visualized in Figure 3.4. It is inf-complete, that is, for any set from \mathbb{N}^2 , one can pick a greatest lower bound. However, the lattice is not sup-complete: For instance, the set $\mathbb{N} \times \{0\}$ has no upper bound. If we take the natural numbers extended with an upper bound ∞ , \mathbb{N}_∞ , as basis, then $(\mathbb{N}_\infty^2, \leq)$ forms a complete lattice.

When working with vectors, we assume standard notation, in particular, for d -dimensional *vector addition* $b + c := (b_1 + c_1, \dots, b_d + c_d)$. By \hat{e}_k , we denote

the *unit vector* where the k -th component is 1 and all other components equal 0. For example, \hat{e}_2 in a 3-dimensional context equals $(0, 1, 0)$. The *zero vector* $(0, \dots, 0)$ is denoted by $\mathbf{0}$.

3.2 The Linear-Time–Branching-Time Spectrum

Van Glabbeek’s two papers on the “linear-time–branching-time spectrum” (1990, 1993) show how all common equivalences can be understood to form a lattice of sublanguages of HML (and a variant of HML for equivalences with silent steps). His hierarchy of equivalences derives from a hierarchy of *notions of observability*.⁶⁴ We will introduce a similar construction: at first, in a generic form, then for the strong spectrum of van Glabbeek (1990).

⁶⁴ In particular, the weak spectrum (van Glabbeek, 1993) makes this concept of notions really formal. Also, we usually implicitly refer to the full version of the strong spectrum (van Glabbeek, 2001).

3.2.1 Spectra as Observability Lattices

We will discuss various equivalence spectra. Van Glabbeek (1990, 1993) already gives two different ones. Let us first introduce an abstract description of such spectra.

Definition 3.5 (Equivalence spectrum). An *equivalence spectrum* $(\mathbf{N}, \leq, \mathcal{O}_{N \in \mathbf{N}})$ consists of

- a lattice of notions of observability, \mathbf{N} , partially ordered by $\leq \subseteq \mathbf{N} \times \mathbf{N}$, and
- corresponding logics $\mathcal{O}_N : 2^{\text{HML}}$ for $N \in \mathbf{N}$.

$\mathcal{O}_{N \in \mathbf{N}}$ must be monotonic, that is: for any two notions $N, M \in \mathbf{N}$, it holds that

$$N \leq M \text{ implies } \mathcal{O}_N \subseteq \mathcal{O}_M.$$

Let us use our new definition to construct a subset lattice as in Example 3.1 to recreate the hierarchy of Figure 3.2.

Example 3.3 (Diamond spectrum). Consider the notions

$$\mathbf{N}^{\text{simple}} := 2^{\{\oplus, \otimes\}},$$

ordered by subset inclusion, and the family of observation languages $\mathcal{O}_{N \in \mathbf{N}}^{\text{simple}}$ given by the family of grammars with some conditional productions:

$$\begin{array}{lcl} \varphi^N & ::= & \top \\ & | & \langle \alpha \rangle \varphi^N \\ & | & \bigwedge_{i \in I} \varphi_i^N \quad \text{if } \oplus \in N \\ & | & \bigwedge_{i \in I} \neg \langle \alpha_i \rangle \top \quad \text{if } \otimes \in N \\ & | & \neg \varphi^N \quad \text{if } \{\oplus, \otimes\} = N. \end{array}$$

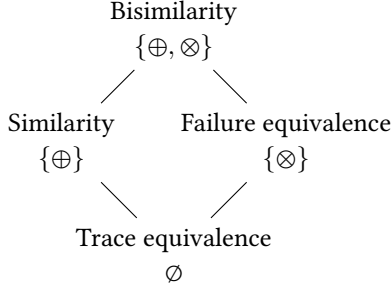


Figure 3.5: Hierarchy of simple notions of equivalence. (From now on, there are no arrows of implication: The lines stand for \leq or \subseteq relationships in the lattice!)

Clearly, $N \subseteq M$ implies $\mathcal{O}_N \subseteq \mathcal{O}_M$ for $N, M \in \mathbf{N}^{\text{simple}}$. We obtain the diamond hierarchy of Figure 3.5. It matches the diamond of Figure 3.2, but this time, the hierarchy is an effect of the ordered notions.

While the incomparable languages of Section 3.1 form no lattice, e.g. $\mathcal{O}_S \cup \mathcal{O}_F \neq \text{HML}$, the *notions* of the present Example 3.3 do form a lattice, as $\{\oplus\} \cup \{\otimes\} = \{\oplus, \otimes\}$. This is one of the reasons why it is convenient to add *notions of observability* as an abstraction layer.

We can also ask what the least notion is to include a specific formula:

Definition 3.6 (Syntactic expressiveness price). In the context of a spectrum $(\mathbf{N}, \leq, \mathcal{O}_{N \in \mathbf{N}})$, the *syntactic expressiveness price* of a formula φ that appears in one of the logics (i.e. $\varphi \in \bigcup_{N \in \mathbf{N}} \mathcal{O}_N$) is defined as

$$\text{expr}(\varphi) := \min\{N \in \mathbf{N} \mid \varphi \in \mathcal{O}_N\}.$$

Note that, given our definition of spectra, the minimum of Definition 3.6 does not need to exist—but it does for the spectra we will be working with.

Thinking of the lattice of notions as a hierarchy of how difficult it is to tell processes apart, we consider this as a kind of “price tag” to put on formulas depending on their syntactic complexity. Higher syntactic complexity allows formula sets of higher expressiveness.

In this view, a trace formula is *cheaper* than a failure formula. Using Example 3.3: The prices differ $\text{expr}(\langle \tau \rangle \langle a \rangle) = \emptyset \subset \{\otimes\} = \text{expr}(\langle \tau \rangle \wedge \{\neg \langle a \rangle\})$, which captures that we need a strictly smaller part of the grammar to construct the trace formula.

The concept of notions of equivalence will allow us to conveniently handle big equivalence hierarchies.

3.2.2 Strong Notions of Observability

Now we can approach the *strong spectrum* by van Glabbeek (1990).⁶⁵ We will encode its notions as a \mathbb{N}_∞ -vector lattice as in Example 3.2. To cover all common behavioral preorders, we use six dimensions, counting certain syntactic features of HML formulas:

1. Modal depth of *observations* ($\langle \alpha \rangle \dots$).
2. Nesting depth of *conjunctions* ($\bigwedge \{ \dots \}$).
3. Maximal modal depth of *deepest positive conjuncts* in conjunctions.
4. Maximal modal depth of the *other positive conjuncts* in conjunctions.
5. Maximal modal depth of *negative conjuncts* in conjunctions.
6. Nesting depth of *negations* ($\neg \dots$).

Definition 3.7 (Strong spectrum). We define the *strong notions of observability* using vectors of extended naturals

$$\mathbf{N}^{\text{strong}} := \mathbb{N}_\infty^6,$$

⁶⁵ “Strong” means that we treat τ like any other action. This traditional naming alludes to the fact that weak notions, which wash away certain differences of internal τ -behavior, are implied by their strong counterparts.

ordered by pointwise comparison of vector components. The *family of strong observation languages* $\mathcal{O}_{N \in \mathbf{N}^{\text{strong}}}^{\text{strong}}$ is given by the parameterized grammar starting from φ^N :⁶⁶

⁶⁶  primrec [Priced_HML.formula_of_price](#)

$$\begin{aligned} \varphi^N &::= \top \\ &| \langle \alpha \rangle \varphi^{N - \hat{e}_1} \\ &| \bigwedge \{ \varphi^{(N - \hat{e}_2) \sqcap (N_3, \infty, \infty, \infty, \infty, \infty)}, \\ &\quad \psi^{(N - \hat{e}_2) \sqcap (\infty, \infty, \infty, N_3, \infty, \infty)}, \psi^{(N - \hat{e}_2) \sqcap (\infty, \infty, \infty, N_3, \infty, \infty)}, \dots \} \\ \psi^N &::= \varphi^{(N \sqcap (N_4, \infty, \infty, \infty, \infty, \infty))} \\ &| \neg \varphi^{(N \sqcap (N_5, \infty, \infty, \infty, \infty, \infty)) - \hat{e}_6} \end{aligned}$$

The productions only exist if the respective recursive invocations are defined on the domain of notions. For instance, $\varphi^N \rightsquigarrow \langle \alpha \rangle \varphi^{N - \hat{e}_1}$ is no valid production for $N = (0, 1, 0, 0, 0, 0)$ because $(-1, 1, 0, 0, 0, 0) \notin \mathbf{N}^{\text{strong}}$.

The number of ψ -conjuncts in $\bigwedge \{ \dots \}$ is free—in particular, there might be none or infinitely many.

Example 3.4 (Formula language). The smallest notion to cover the failure formula of Section 3.1.2 would be $(2, 1, 0, 0, 1, 1)$, that is,

$$\langle \tau \rangle \bigwedge \{ \neg \langle a \rangle \top \} \in \mathcal{O}_{(2,1,0,0,1,1)}^{\text{strong}}.$$

This is because the formula has two levels of modal observations, where the inner one is negated. The negation is wrapped in a conjunction with no positive conjuncts. A visualization for how $\text{expr}^{\text{strong}}(\langle \tau \rangle \bigwedge \{ \neg \langle a \rangle \top \}) = (2, 1, 0, 0, 1, 1)$ (according to Definition 3.6) comes together is given in Figure 3.6. Formally, the reason is that the following derivation is optimal:

$$\begin{aligned} \varphi^{(2,1,0,0,1,1)} &\rightsquigarrow \langle \tau \rangle \varphi^{(1,1,0,0,1,1)} \\ &\rightsquigarrow \langle \tau \rangle \bigwedge \{ \psi^{(1,0,0,0,1,1)} \} \\ &\rightsquigarrow \langle \tau \rangle \bigwedge \{ \neg \varphi^{(1,0,0,0,1,0)} \} \\ &\rightsquigarrow \langle \tau \rangle \bigwedge \{ \neg \langle a \rangle \varphi^{(0,0,0,0,1,0)} \} \\ &\rightsquigarrow \langle \tau \rangle \bigwedge \{ \neg \langle a \rangle \top \}. \end{aligned}$$

If we reexamine the grammar of \mathcal{O}_F in Definition 3.3, we notice that the formulas it can produce almost match those of $\mathcal{O}_{(\infty,1,0,0,1,1)}^{\text{strong}}$ of the strong spectrum in Definition 3.7. The latter generates the φ_F -grammar in Figure 3.9. As Lemma 3.3 will show, both languages are equally distinctive (Definition 2.15).

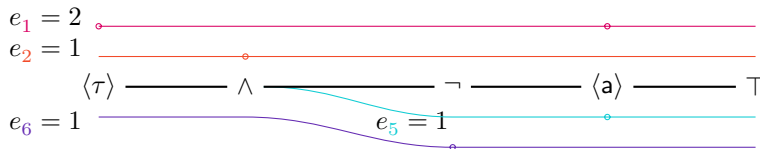


Figure 3.6: Pricing formula $\langle \tau \rangle \bigwedge \{ \neg \langle a \rangle \top \}$ with syntactic expressiveness of $(2, 1, 0, 0, 1, 1)$.

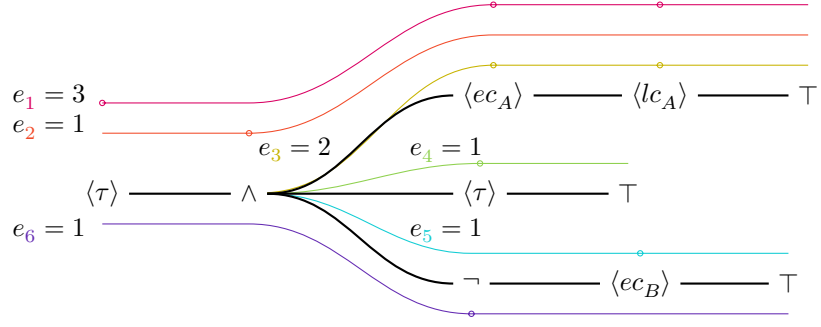


Figure 3.7: Pricing formula $\langle \tau \rangle \wedge \{ \langle ec_A \rangle \langle lc_A \rangle \top, \langle \tau \rangle \top, \neg \langle ec_B \rangle \top \}$ with syntactic expressiveness $(3, 1, 2, 1, 1, 1)$.

An example for the pricing of a more complex tree-like formula is given in Figure 3.7.

Remark 3.1 (Explicit formula prices). In my papers (Bisping et al., 2022; Bisping, 2023b; Bisping & Jansen, 2024), the syntactic expressiveness price of formulas expr is explicitly defined, instead of using a family of grammars. In this thesis, expr is indirectly defined through Definition 3.6.

The expressiveness price $\text{expr}^{\text{strong}}$ of formulas in the strong spectrum (Definition 3.7) can be recursively characterized as described in Bisping (2023b) (we will drop the “strong” for readability):

$$\begin{aligned}
 \text{expr}(\top) &= 0 \\
 \text{expr}(\langle \alpha \rangle \varphi) &= \hat{e}_1 + \text{expr}(\varphi) \\
 \text{expr}(\neg \varphi) &= \hat{e}_6 + \text{expr}(\varphi) \\
 \text{expr}\left(\bigwedge_{i \in I} \psi_i\right) &= \hat{e}_2 + \sup \left(\left\{ \begin{array}{c} 0 \\ 0 \\ \sup_{i \in \text{Pos}} (\text{expr}(\psi_i))_1 \\ \sup_{i \in \text{Pos} \setminus R} (\text{expr}(\psi_i))_1 \\ \sup_{i \in \text{Neg}} (\text{expr}(\psi_i))_1 \\ 0 \end{array} \right\} \cup \{ \text{expr}(\psi_i) \mid i \in I \} \right) \\
 \text{Neg} &:= \{ i \in I \mid \exists \varphi'_i. \psi_i = \neg \varphi'_i \} \\
 \text{Pos} &:= I \setminus \text{Neg} \\
 R &:= \begin{cases} \emptyset & \text{if } \text{Pos} = \emptyset \\ \{r\} & \text{for some } r \in \text{Pos} \text{ with } (\text{expr}(\psi_r))_1 \text{ max. for Pos.} \end{cases}
 \end{aligned}$$

Note that there is a minor divergence from Bisping (2023b) in that the present thesis prices $\text{expr}(\top)$ at 0 instead of \hat{e}_2 . This change is done to align the pricing to the one we will need for the weak spectrum in later chapters.

Clearly, the explicit pricing of formulas is more versatile for the implementation direction we are aiming at. The grammar view, on the other hand, is particularly nice to see how our HML sublanguages will align to games.

The strong spectrum of Definition 3.7 covers the notions of behavioral equivalence we have discussed so far. (What grammars due to the coordinates look like is listed in Figure 3.9 and Figure 3.10.)

Lemma 3.3 (Spectrum characterization). *Traces, simulation, bisimulation and failures are covered as follows:*

1. The observation language $\mathcal{O}_{(\infty,0,0,0,0,0)}^{\text{strong}}$ exactly matches the characterization of traces \mathcal{O}_T from Definition 3.1 and thus characterizes trace pre-order.⁶⁷
2. The observation language $\mathcal{O}_{(\infty,\infty,\infty,\infty,0,0)}^{\text{strong}}$ exactly matches the characterization of simulation observations \mathcal{O}_S from Definition 3.1 and thus characterizes simulation.⁶⁸
3. The observation language $\mathcal{O}_{(\infty,\infty,\infty,\infty,\infty,\infty)}^{\text{strong}}$ matches HML in distinctiveness and thus characterizes bisimilarity.
4. The observation language $\mathcal{O}_{(\infty,1,0,0,1,1)}^{\text{strong}}$ matches failure observations \mathcal{O}_F of Definition 3.3 in distinctiveness.

⁶⁷  theorem [Priced_Spectrum.lts.traces_priced_characterization](#)

⁶⁸  theorem [Priced_Spectrum.lts.simulation_priced_characterization](#)

Proof.

- Claim (1) for traces is trivial.
- Claim (2) for simulation is trivial, given flattening of conjunctions.
- For claim (3) of bisimilarity, observe that $\mathcal{O}_{[B]} \subseteq \mathcal{O}_{(\infty,\infty,\infty,\infty,\infty,\infty)}^{\text{strong}}$ by examining its grammar in Definition 2.16. As $\mathcal{O}_{[B]}$ already has complete HML distinctiveness by Lemma 2.7, so must its superlogic $\mathcal{O}_{(\infty,\infty,\infty,\infty,\infty,\infty)}^{\text{strong}}$.
- For claim (4) of failures, we must note that $\mathcal{O}_{(\infty,1,0,0,1,1)}^{\text{strong}}$ also contains failure observations that end in conjunctions with a $\neg\top$ -conjunct. But these are trivially false and thus cannot add to distinctiveness. \square

So far, we have only established that the six-dimensional spectrum covers the notions that the diamond of Example 3.3 has already covered—in a more complicated way. The extra dimensions will pay off in the next subsection.

3.2.3 The Strong Linear-Time–Branching-Time Spectrum

Using the six dimensions of Definition 3.7, we can assign coordinates to all other common notions of the strong linear-time–branching-time spectrum.

Definition 3.8 (Strong linear-time–branching-time spectrum). Coordinates with respect to the notions of Definition 3.7 for the common notions of behavioral equivalence and preorder in the strong linear-time–branching-time spectrum are given in Figure 3.8.⁶⁹

⁶⁹ When writing vectors in labels and figures, we omit the parentheses (...) for readability.

The coordinates of Definition 3.8 define a hierarchy of modal languages. Figure 3.9 and Figure 3.10 list the grammars that are described through the coordinates in interplay with Definition 3.7. The defining aspects of each grammar are marked in blue. The conjunction productions with “...” are to be read in the sense that they allow arbitrary many conjuncts; in particular, empty and

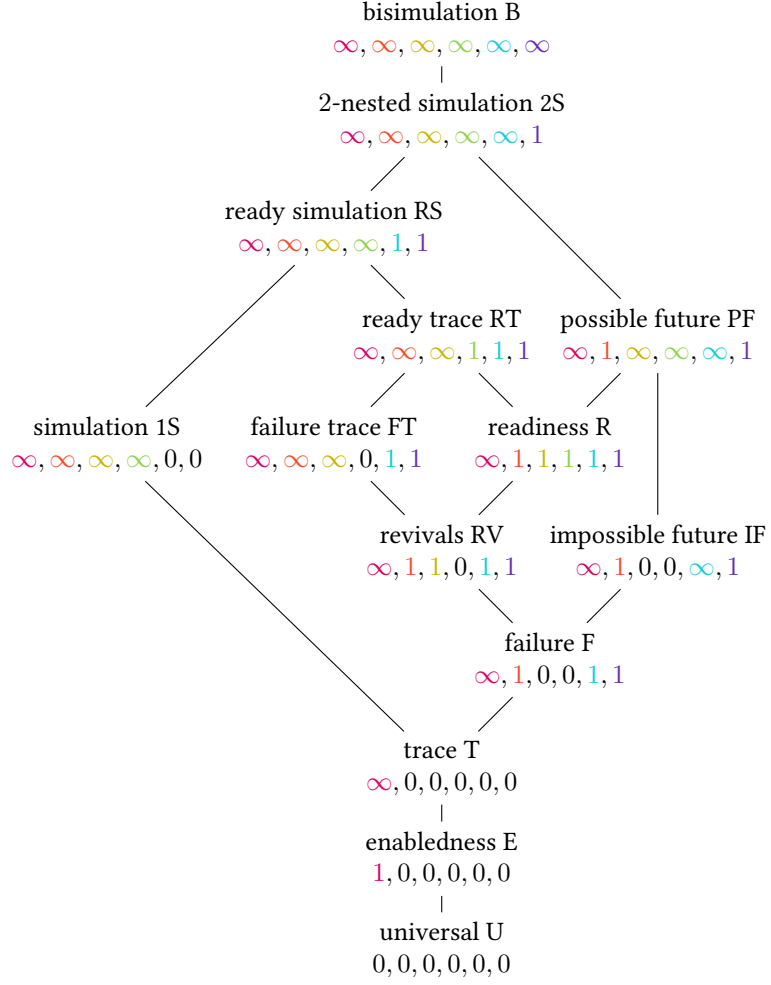


Figure 3.8: Hierarchy of common equivalences/preorders ordered by observability coordinates.

infinitary conjunctions are possible. If a non-terminal appears only once in the conjunction production, at most one such subformula is allowed.

For the rest of the thesis, we will take the equivalences as defined by the coordinates as canonical. But, of course, it is natural to ask whether the equivalences thus defined correspond to previous definitions in this thesis and the literature. Lemma 3.3 already establishes that the coordinates of traces, simulation, failures, and bisimulation match the common definitions.

Mattes (2024) proves in Isabelle/HOL that our coordinate system spectrum matches the distinctiveness of the modal characterizations by van Glabbeek (2001).⁷⁰ This entails a lot of fine print, which we will not reproduce here.

The grammars already contain a trick to handle *equivalences of decorated traces*, which we should discuss as it will be mirrored in game rules later on:

⁷⁰ To be precise, Mattes (2024) works with the version of the spectrum from Bisping (2023b). The only real difference is that, there, \top is priced at 1 conjunction, while the present spectrum prices it at 0, cf. Remark 3.1.

Universal, U at $(0, 0, 0, 0, 0, 0)$:

$$\varphi_U ::= \top$$

Enabledness, E at $(1, 0, 0, 0, 0, 0)$:

$$\varphi_E ::= \langle \alpha \rangle \varphi_U \mid \varphi_U$$

Traces, T at $(\infty, 0, 0, 0, 0, 0)$:

$$\varphi_T ::= \langle \alpha \rangle \varphi_T \mid \varphi_U$$

Failures, F at $(\infty, 1, 0, 0, 1, 1)$:

$$\begin{aligned} \varphi_F &::= \langle \alpha \rangle \varphi_F \mid \bigwedge \{ \psi_F, \psi_F, \dots \} \\ \psi_F &::= \neg \varphi_E \mid \top \end{aligned}$$

Revivals, RV at $(\infty, 1, 1, 0, 1, 1)$:

$$\varphi_{RV} ::= \langle \alpha \rangle \varphi_{RV} \mid \bigwedge \{ \varphi_E, \psi_F, \psi_F, \dots \}$$

Readiness, R at $(\infty, 1, 1, 1, 1, 1)$:

$$\begin{aligned} \varphi_R &::= \langle \alpha \rangle \varphi_R \mid \bigwedge \{ \psi_R, \psi_R, \dots \} \\ \psi_R &::= \varphi_E \mid \psi_F \end{aligned}$$

Failure trace, FT at $(\infty, 1, \infty, 0, 1, 1)$:

$$\varphi_{FT} ::= \langle \alpha \rangle \varphi_{FT} \mid \bigwedge \{ \varphi_{FT}, \psi_F, \psi_F, \dots \}$$

Ready trace, RT at $(\infty, 1, \infty, 1, 1, 1)$:

$$\varphi_{RT} ::= \langle \alpha \rangle \varphi_{RT} \mid \bigwedge \{ \varphi_{RT}, \psi_R, \psi_R, \dots \}$$

Figure 3.9: Grammars induced by coordinates for strong linear-time notions of equivalence.

The idea behind *failure traces* is that one observes a trace of actions and *at each step* a set of actions that are *disabled*. *Ready traces* work similarly, but with disabled and enabled actions in between.

Example 3.5 (Failure traces). Consider the following processes, which are slightly deeper variants of the situation we encountered with Q and troll process T in Example 2.7.

$$\begin{aligned} Q' &:= \tau.(a.a + b.b) \\ T'_{aa} &:= Q' + \tau.a.a \\ T'_a &:= Q' + \tau.a \end{aligned}$$

Impossible futures, IF at $(\infty, 1, 0, 0, \infty, 1)$:

$$\begin{aligned}\varphi_{\text{IF}} &::= \langle \alpha \rangle \varphi_{\text{IF}} \mid \bigwedge \{ \psi_{\text{IF}}, \psi_{\text{IF}}, \dots \} \\ \psi_{\text{IF}} &::= \neg \varphi_{\text{T}} \mid \top\end{aligned}$$

Possible futures, PF at $(\infty, 1, \infty, \infty, \infty, 1)$:

$$\begin{aligned}\varphi_{\text{PF}} &::= \langle \alpha \rangle \varphi_{\text{PF}} \mid \bigwedge \{ \psi_{\text{PF}}, \psi_{\text{PF}}, \dots \} \\ \psi_{\text{PF}} &::= \varphi_{\text{T}} \mid \psi_{\text{IF}}\end{aligned}$$

Simulation, 1S at $(\infty, \infty, \infty, \infty, 0, 0)$:

$$\varphi_{1\text{S}} ::= \langle \alpha \rangle \varphi_{1\text{S}} \mid \bigwedge \{ \varphi_{1\text{S}}, \varphi_{1\text{S}}, \dots \}$$

Ready simulation, RS at $(\infty, \infty, \infty, \infty, 1, 1)$:

$$\begin{aligned}\varphi_{\text{RS}} &::= \langle \alpha \rangle \varphi_{\text{RS}} \mid \bigwedge \{ \psi_{\text{RS}}, \psi_{\text{RS}}, \dots \} \\ \psi_{\text{RS}} &::= \psi_{\text{F}} \mid \varphi_{\text{RS}}\end{aligned}$$

2-nested simulation, 2S at $(\infty, \infty, \infty, \infty, \infty, 1)$:

$$\begin{aligned}\varphi_{2\text{S}} &::= \langle \alpha \rangle \varphi_{2\text{S}} \mid \bigwedge \{ \psi_{2\text{S}}, \psi_{2\text{S}}, \dots \} \\ \psi_{2\text{S}} &::= \neg \varphi_{1\text{S}} \mid \varphi_{2\text{S}}\end{aligned}$$

Bisimulation, B at $(\infty, \infty, \infty, \infty, \infty, \infty)$:

$$\begin{aligned}\varphi_{\text{B}} &::= \langle \alpha \rangle \varphi_{\text{B}} \mid \bigwedge \{ \psi_{\text{B}}, \psi_{\text{B}}, \dots \} \\ \psi_{\text{B}} &::= \neg \varphi_{\text{B}} \mid \varphi_{\text{B}}\end{aligned}$$

Figure 3.10: Grammars induced by coordinates for strong branching-time notions of equivalence.

T'_{aa} and T'_a are simulation equivalent. But they are distinguished by the failure-trace observation $\langle \tau \rangle \bigwedge \{ \langle a \rangle \langle a \rangle, \neg \langle b \rangle \}$. The intuition is that we can observe the trace $\tau a a$ together with the fact that b is impossible after a first step of T'_{aa} , but not of T'_a .

The formula pricing works out, that is, $\langle \tau \rangle \bigwedge \{ \langle a \rangle \langle a \rangle, \neg \langle b \rangle \} \in \mathcal{O}_{(3,1,2,0,1,1)}^{\text{strong}} \subseteq \mathcal{O}_{\text{FT}}^{\text{strong}}$. The important point here is that we can allow *one* arbitrarily deep positive conjunct but prevent any other, which would undermine the *trace-likeness* of the observation.

Example 3.5 shows how it pays off that the spectrum grammar (Definition 3.7) has a more lenient pricing for one of the positive conjuncts. We will refer to the deep positive conjunct of a conjunction as the *revival*, because revivals equivalence (Reed et al., 2007) is the minimal notion with this feature. It also matters for ready traces, as seen in the ready trace formula of Figure 3.7.

3.2.4 Any Questions?

There are a few standard questions that come to mind for people who are familiar with the various spectra of equivalence when seeing Figure 3.8. The following remarks address these points.

Remark 3.2 (Selection of notions). At the core, we treat the same notions as van Glabbeek (1990). But we feature a slightly more modern selection.

Our spectrum additionally includes strong versions of *impossible futures* (Voorhoeve & Mauw, 2001) and *revivals* (Reed et al., 2007; Roscoe, 2009) as equivalences whose relevance has only been noted after the publication of van Glabbeek (2001).

On the other hand, we glimpse over completed trace, completed simulation, and possible worlds observations like Kučera & Esparza (1999), who studied properties of “good” observation languages. These notions would need a different HML grammar, featuring exhaustive conjunctions $\bigwedge_{\alpha \in Act} \varphi_\alpha$, where the φ_α are deactivated actions for completed traces, and more complex trees for possible worlds.

Remark 3.3 (Synonymous coordinates). For many of the logics in Figure 3.8, there are multiple coordinates that characterize the same logic. For instance, due to the second dimension (conjunctions) being set to 0 for traces T , the higher dimensions do not matter and any coordinate $N = (\infty, 0, N_3, N_4, N_5, N_6)$ will lead to the same observation language $\mathcal{O}_N^{\text{strong}} = \mathcal{O}_T^{\text{strong}}$.

Ruling out such equalities in the design of the lattice would be quite tedious. Luckily, Definition 3.5 only demands $N \leq M$ to imply $\mathcal{O}_N \subseteq \mathcal{O}_M$, and not the converse.

Indeed, Figure 3.8 always selects the least coordinate to characterize a sublogic, in order for domination of coordinates in the figure and entailment between behavioral preorders to coincide.

At the same time, some notions could be characterized by strictly smaller coordinates due to sublogics of matching expressiveness. In particular, all $(\infty, \infty, N_3, N_4, \infty, \infty)$ would characterize bisimilarity due to the unbounded possibility of “hiding” positive conjuncts in double negations. This observation will become relevant again later for varying abstractions of bisimilarity in the spectrum of weak equivalences in Section 6.3.2.

Remark 3.4 (Other coordinates). We have singled out a handful of coordinates. Many other coordinates will still correspond to distinct equivalences. For instance, we could consider $N^{2T} = (2, 0, 0, 0, 0, 0)$, preordering states that cannot be distinguished by traces up to a length of 2. But it is difficult to make a case for such a “notion of equivalence,” which washes away differences of future behavior after exactly two steps.

Some kinds of depth-bounded families, however, are common in the literature to approximate bisimilarity and can also be placed in our spectrum:

- k -step bisimilarity: $(k, \infty, \infty, \infty, \infty, \infty)$ is a depth- k approximation of

bisimilarity that sometimes appears in its fixed point characterizations, for instance in Aceto et al. (2007, Section 4.3).

- k -stratified bisimilarity: $(\infty, k, \infty, \infty, \infty, \infty)$ would be what Milner (1989, Section 10.4) calls “stratification of bisimilarity,” also appearing as “ $(k + 1)$ -nested trace equivalence” in Aceto et al. (2004).
- k -nested similarity: $(\infty, \infty, \infty, \infty, \infty, k - 1)$ for $k > 1$ defines a spectrum of modal quantifier alternation depth between similarity and bisimilarity.

Remark 3.5 (Alternate dimensions). In principle, one can choose different dimensions to characterize the strong linear-time–branching-time spectrum. (Indeed, Bisping et al. (2022), Bisping (2023b) and this thesis all use slightly different dimensions.)

Naturally, one may ask whether the same spectrum can be characterized with fewer than six dimensions. As some of the preceding examples also show, the dimensions we chose are not entirely orthogonal.

If we restrict our focus to a grid of ternary entries $\{0, 1, \infty\}$, we can be sure to need at least five dimensions: With four dimensions, the height of the lattice is nine, that is, the maximal number of nodes on increasing paths between $(0, 0, 0, 0)$ and $(\infty, \infty, \infty, \infty)$. But the hierarchy of Figure 3.8 has height ten!

With a grid of five ternary dimensions, we can recreate the hierarchy of Figure 3.8, and “hard-code” the logics for coordinates, a bit as it happens in Example 3.3. On the other hand, this would align less nicely with syntactic features of Hennessy–Milner logic.

Remark 3.6 ((In-)finitary variants). One can introduce more dimensions to the spectrum with respect to the possibility of infinitary observations. Our choice focuses on natural and most common versions of the equivalences, in particular: similarity and bisimilarity with unbounded (infinitary) branching and trace-like notions with finitary depth. Notions in Figure 3.8 correspond precisely to those without superscripts in the infinitary linear-time–branching-time spectrum of van Glabbeek (2001, Figure 9).

3.2.5 Non-Intersectionality

The strong spectrum of Definition 3.8 is much richer than the diamond spectrum from Example 3.3. Still, its observation languages form no lattice. For instance, the lines of simulation and failures join at ready simulation—and their coordinates as well $(\infty, \infty, \infty, \infty, 0, 0) \sqcup (\infty, 1, 0, 0, 1, 1) = (\infty, \infty, \infty, \infty, 1, 1)$. But $\mathcal{O}_S^{\text{strong}} \cup \mathcal{O}_F^{\text{strong}} \neq \mathcal{O}_{RS}^{\text{strong}}$ and this makes a difference:

Example 3.6 (Simulation + failures \neq ready-simulation). Consider the CCS processes $a.(a.b + a)$ and $a.a.b + a.a$. They cannot be told apart by $\mathcal{O}_S^{\text{strong}}$ or $\mathcal{O}_F^{\text{strong}}$ and thus are simulation and failure equivalent (and moreover even ready-trace equivalent).

Still, the formula $\langle a \rangle \wedge \{ \langle a \rangle \wedge \neg \langle b \rangle \}, \langle a \rangle \langle b \rangle \} \in \mathcal{O}_{\text{RS}}^{\text{strong}}$ distinguishes the first process from the second. Therefore, the processes are not ready-simulation equivalent.

What Example 3.6 shows is that one cannot prove two states to be ready-simulation-equivalent by showing that they are equated by simulation and failures:

$$\sim_S \cap \sim_F \not\subseteq \sim_{\text{RS}}.$$

The relationship between the characterized equivalences is *non-intersectional*.

In general, multiple preorders may relate two states without this entailing a stronger equivalence. So the question “Which equivalence from a spectrum relates two states?” is too simple—one has to ask in *plural*: “Which equivalences relate two processes?”

This plural motivates the *spectroscopy problem*.

3.3 Spectroscopy

Now that we have a formal way of describing equivalence spectra, we can make formal the *spectroscopy problem*—the core topic of this thesis. We will also collect first thoughts on its complexity.

3.3.1 The Spectroscopy Problem

The problem has originally been introduced in Bisping et al. (2022) as the “abstract observation preorder problem” with respect to modal characterizations of the strong spectrum. We here reintroduce it in a more generic form.

i Problem 1: Spectroscopy problem

In the context of a transition system \mathcal{S} and a spectrum $(\mathbb{N}, \leq, \mathcal{O}_{\mathbb{N}})$, the *spectroscopy problem* asks:

Input States p and q .

Output Maximal set of notions $\mathbb{N}_{p,q} \subseteq \mathbb{N}$, such that $p \preceq_{\mathcal{O}_N} q$ for each $N \in \mathbb{N}_{p,q}$.

Intuitively, the problem is about finding all the ways in which processes can be related, beyond the black-and-white of the bisimilarity problem. We aim to *split up* information on possible distinctions, analogously to a prism with light (Figure 3.11). Given that the equivalence hierarchies are commonly referred to as *spectra*, it is natural to borrow the name *spectroscopy* from physical experiments.

Example 3.7 (Warm-up spectroscopy). In Example 2.4, we have noticed that it is easy to distinguish $q_{ab} \not\preceq_T p_a$ through the 1-symbol-trace b . In HML, the difference is expressed as $\langle b \rangle \top$.

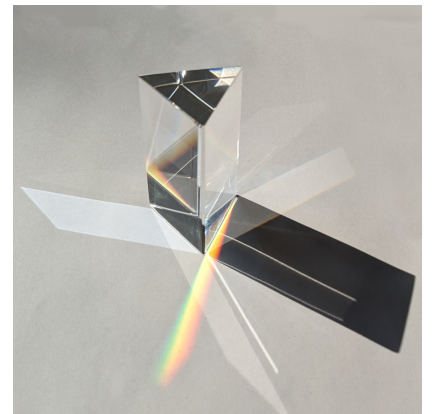


Figure 3.11: A prism revealing the spectrum of sun light.

The solution to the strong spectroscopy problem on q_{ab} and p_a is

$$N_{q_{ab}, p_a}^{\text{strong}} = \{N \in N^{\text{strong}} \mid N_1 < 1\}.$$

All notions thus characterized correspond to universal equivalence U in distinctiveness. In other words, the states are not equivalent with respect to any notion where states are told apart at all. They are *as different as possible*, in our metric.

Example 3.8 (The spectrum of trolled philosophers). For the “trolled philosophers” of Example 2.7, we have determined that the systems are simulation-preordered, but not bisimilar, that is, $Q \preceq_S T$, but $Q \not\preceq_B T$. The first fact implies $Q \preceq_T T$.

But what about other notions from the strong spectrum of Section 3.2.3? Besides similarity, there might well be incomparable or finer notions that also preorder Q to T !

The solution to the spectroscopy problem on Q and T is

$$N_{Q,T}^{\text{strong}} = \{N \in N^{\text{strong}} \mid (2, 2, 0, 0, 2, 2) \not\preceq N\}.$$

A minimal formula to distinguish Q from T with coordinate $(2, 2, 0, 0, 2, 2)$ would be $\bigwedge \{\neg \langle \tau \rangle \wedge \{\neg \langle a \rangle \top\}\}$. (The following chapters will reveal how to reliably arrive at this knowledge, in particular, the minimality.)

Figure 3.12 shows how the distinction is above the distinction spaces of most notions we named.

Because the coordinate of 2-nested simulation, $2S = (\infty, \infty, \infty, \infty, \infty, 1)$ is not above $(2, 2, 0, 0, 2, 2)$, we arrive at $Q \preceq_{2S} T$, which implies *all* preorders of Figure 3.8 except for bisimilarity.

Note that we have expressed $N_{Q,T}^{\text{strong}}$ through a negation (“ $(2, 2, 0, 0, 2, 2) \not\preceq N$ ”). The reason is that a positive description is usually unwieldy. In this (comparably easy) case, we could for example list the half-spaces undercutting the cheapest distinction, and this would read: $N_{Q,T}^{\text{strong}} = (\{0, 1\} \times N_\infty^5) \cup (N_\infty \times \{0, 1\} \times N_\infty^4) \cup (N_\infty^4 \times \{0, 1\} \times N_\infty) \cup (N_\infty^5 \times \{0, 1\})$.

Technically, it is convenient to not compute $N_{p,q}$ directly. Rather we aim to construct the *Pareto front* of minimal notions that do not hold, $\text{Min}(N \setminus N_{p,q})$. The Pareto front serves as a unique representation, from which $N_{p,q}$ can be constructed as complement of the upward closure $N \setminus \uparrow \text{Min}(N \setminus N_{p,q})$. Pareto fronts form *anti-chains* and appear naturally in optimization problems.

All spectra we are concerned with are well-quasi ordered, which means that each $\text{Min}(N \setminus N_{p,q})$ must be finite in size (Kruskal, 1972) and thus “more handy” than the full sets $N_{p,q}$ or $N \setminus N_{p,q}$.

So, effectively, we will be asking: What are the minimal notions to distinguish p from q —and then often talk about the converse: The most-fitting notions to preorder or equate the states. Everything else is implied.

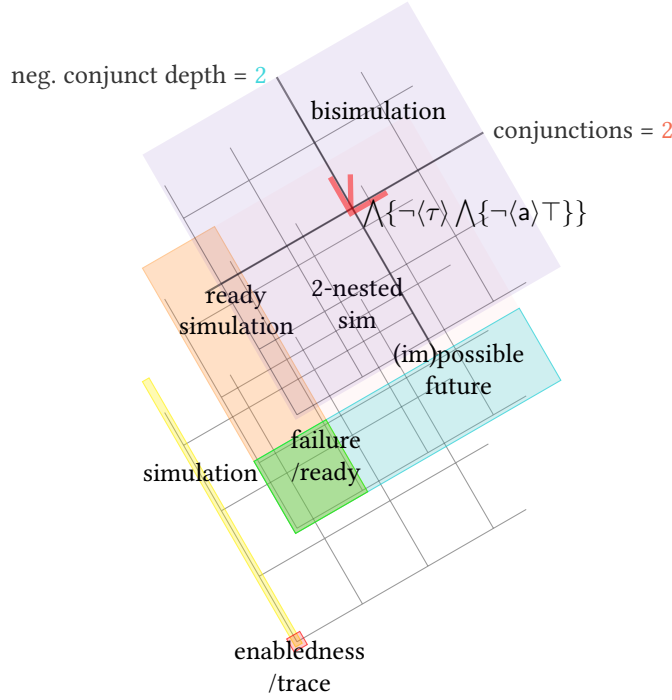


Figure 3.12: Cross section of the strong spectrum showing the dimensions conjunctions, negative conjunct depth, and negation depth. All preorders that are not hit by the red mark at $(2, 2, 2)$ hold in Example 3.8.

Remark 3.7 (Infinitely many notions). The spectrum as formulated in Definition 3.7 contains infinitely many notions. This would be a problem in a naive solution to Problem 1 that relies on deciding equivalences individually, but will be no problem for us. Where we, however, assume finiteness for algorithms, is in the state-space size of \mathcal{S} .

3.3.2 Spectroscopy as Abstract Subtraction

Another way of viewing the spectroscopy problem is that we aim to compute an abstracted kind of *difference* between programs.

Definition 3.9 (Observations and difference). On a transition system \mathcal{S} , the possible observations of a state, $\llbracket \cdot \rrbracket^{\bullet} : \mathcal{P} \rightarrow 2^{\text{HML}}$, are defined as:

$$\llbracket p \rrbracket^{\bullet} := \{ \varphi \in \text{HML} \mid p \in \llbracket \varphi \rrbracket \}.$$

The *difference* between p and q is defined as:

$$\Delta(p, q) := \llbracket p \rrbracket^{\bullet} \setminus \llbracket q \rrbracket^{\bullet}.$$

$\Delta(p, q)$ expresses the set of observations one could make of p that one cannot make of q . This set will be empty when the states are bisimilar, or infinite,

otherwise.

With this notation, we could rephrase how preorders derive from HML subsets in Definition 2.13:

Proposition 3.1. *Two states p and q are preordered with respect to a sublogic $\mathcal{O} \subseteq \text{HML}$:*

$$p \preceq_{\mathcal{O}} q \iff \Delta(p, q) \cap \mathcal{O} = \emptyset.$$

The spectroscopy problem then is about computing some abstraction Δ_A such that $N \in \Delta_A(p, q)$ precisely if $\Delta(p, q) \cap \mathcal{O}_N \neq \emptyset$. $N_{p,q}$ plays the role of $\Delta_A(p, q)$.

This falls in line with our observations about the \leq -game on numbers in Example 2.14 and (bi-)simulation games. Comparison is inherently linked to subtraction.

3.3.3 Complexities

What complexities to expect when deciding spectroscopy problems on finite systems? Details depend, of course, on the specific spectrum and flavor of Hennessy–Milner logic we are concerned with. Still, solving the spectroscopy problem cannot be easier than solving the covered individual equivalence problems.

To get a first idea, let us examine the complexities of common equivalence checking problems in the strong spectrum. The rule of thumb is that trace-like equivalences are PSPACE-complete and bisimilarities are P-complete (Balczár et al., 1992; Hüttel & Shukla, 1996; Kanellakis & Smolka, 1983).

Bisimilarity finds itself in a *valley* of tractability, if we look at a cross section through the equivalence spectrum as in Figure 3.13. The best known bisimilarity algorithms for finite-state transition systems take $O(|\rightarrow| \log |\mathcal{P}|)$ time. They usually employ partition refinement (e.g. Valmari, 2009), deriving from Paige & Tarjan (1987).

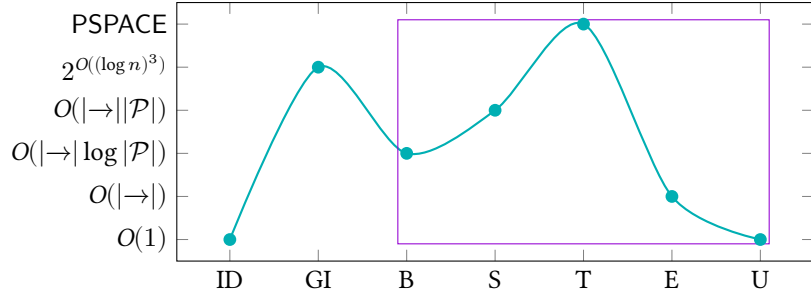


Figure 3.13: Bisimilarity's complexity valley.

For coarser simulation-like equivalences, the best known algorithms need $O(|\rightarrow||\mathcal{P}|)$ time (Ranzato & Tapparo, 2010).⁷¹

⁷¹ Or $O(|\rightarrow||S_{/\sim_S}|)$ to name the bound as Ranzato & Tapparo (2010) present it.

The finer graph-isomorphism equivalence (Definition 2.8) again is harder with the best known solution (Babai, 2016) in quasi-polynomial time $2^{O((\log n)^3)}$.

There are few strict hardness results at this level of granularity. So, better time complexities for graph isomorphism, bisimilarity, and similarity are conceivable (albeit improbable). Groote et al. (2023) show that at least partition-refinement algorithms for bisimilarity cannot do better than $O(|\rightarrow| \log |\mathcal{P}|)$. In a recent preprint, Groote & Martens (2024) establish that similarity is strictly more complex than bisimilarity.

The trivial equivalences at the end of the cross section, identity ID and universal equivalence U, can be solved directly. Enabledness equivalence E can as well be computed quite quickly by just comparing outgoing transitions.

In this thesis, we solve the spectroscopy problem for the strong and weak spectrum. So, we must be at least as complex as the equivalences between bisimilarity and universal equivalence, boxed in Figure 3.13. Consequently, the spectroscopy problem for the standard equivalence spectra is PSPACE-hard.

3.4 Discussion

In this chapter, we have formalized how to handle *spectra of equivalence* (Idea 4), and instantiated the approach to the strong spectrum of van Glabbeek (2001). From there, we have introduced the *spectroscopy problem*, which asks for notions to preorder compared states (Idea 1).

The shifted problem. By formulating the problem in terms of a lattice over \mathbb{N}_∞ -vectors, the family of qualitative strong preorder/equivalence problems becomes a single quantitative problem: The spectroscopy problem for the strong spectrum. As we will see, the perspective of one quantitative problem is more insightful than the view of loosely-related, isolated equivalence problems.

We have already laid the groundwork to shift the semantic question of equivalence into a syntactic question of the shape of distinguishing formulas.

Prior publications. The core ideas of this section have already been explored in Bisping et al. (2022) and Bisping (2023b) for the strong spectrum. However, in my prior work the expressiveness prices played a more crucial role. Here, we instead opted for a parameterized grammar to define notions and their observations \mathcal{O}_N . In this grammar, we count \top as part of the 0-notion. These are mostly superficial changes to streamline the following presentation. We have shown that traces, failures, simulation and bisimulation equivalence as defined by the notion coordinates match their textbook definitions. Mattes (2024) addresses all notions in Isabelle/HOL.

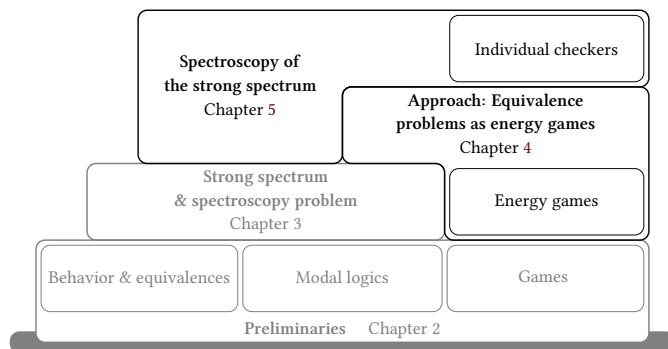
What's next? So far, we have a problem and confidence that its solution conveys information about equivalences of the strong spectrum. Section 3.3.3

has established that spectroscopy complexity must be at least PSPACE on the strong spectrum.

However, there is a polynomial-time-easy fragment of the spectrum around bisimilarity and (ready) similarity. In Part II, comprising the next two chapters, we will first solve the *spectroscopy problem for this P-easy slice* and then extend to the whole strong spectrum. After that, we will also consider the weak spectrum in Part III.

Part II

Generalized Equivalence Checking



4 Approach: Equivalence Problems as Energy Games

Time to get real. This chapter will demonstrate my core approach by solving the spectroscopy problem in an *easy* instance.

It will be easy in two regards: Firstly, complexity-wise—we narrow our focus on equivalences of polynomial-time, the “P-easy ones.” Secondly, conceptually—we just use the bisimulation game of Chapter 2, almost directly!

We already know from Theorem 2.3 that winning attacker strategies in the bisimulation game correspond to distinguishing formulas, and that these can be computed quite straightforwardly. All we need on top of this is a way of quantifying the amount of syntactic expressiveness in these formulas *during the game*. For this, we employ the first core idea of this chapter:

i Idea 6: Quantitative games for quantitative problems

Spectra of equivalence problems can be encoded as energy games.

Energy games are games in which players have limited resources that can be used up or recharged during moves. Players running out of a resource lose the game. After introducing such games in Section 4.1, we will prove in Section 4.2 that several core equivalences can be characterized through attacker energy budgets where the defender wins. Energies in the game will correspond to notions in the spectrum. For this, we will add three-dimensional energies to the bisimulation game.

The second core idea is how to compute these winning budgets for players:

i Idea 7: Computing cheapest wins

Attacker’s winning budgets in energy reachability games can be computed by a generalized shortest paths algorithm.

Section 4.3 provides an algorithm to solve a range of energy-game-like quantitative problems as long as the energy updates can be *undone* through a *Galois connection*—a generalization of invertibility on monotonic functions.

Related publications. This chapter revolves around the ideas conveyed by my talk “Deciding all behavioral equivalences at once through energy games” at D-CON’25 (Augsburg, 2025-03), and by its predecessors given at LFCS University of Edinburgh (2023-03), Highlights’23 (Kassel, 2023-07), and ISCAS Beijing (2024-09). It can be thought of as the *core* of Bisping (2023b), and also of this thesis itself.

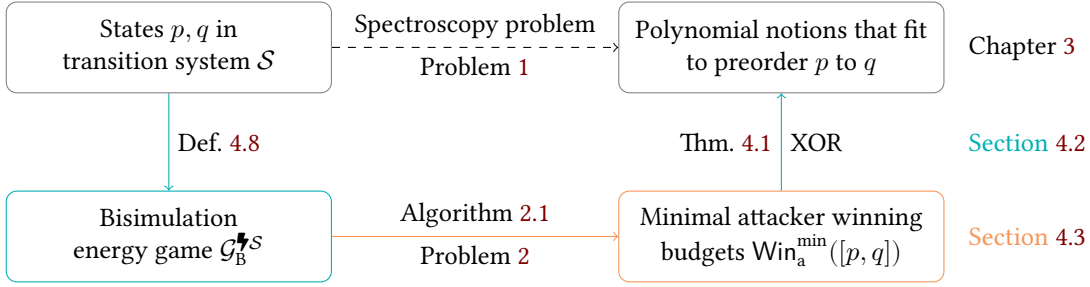


Figure 4.1: How we will employ energy games to solve the spectroscopy problem on an easy spectrum.

By combining the two contributions, we arrive at a polynomial-time solution of the spectroscopy problem for the P-easy slice of the strong spectrum.

Effectively, we adapt the game framework of Figure 2.18 to not treat one equivalence, but a spectrum of equivalences. This approach is summarized in Figure 4.1.

4.1 Energy Games

Energy games extend games as discussed in Section 2.4 with resources of the players, called *energies*. We will focus on reachability games with an energy-bounded attacker. In most publications, energy games update vector-valued energy levels by vector addition or subtraction. We work with more general *monotonic energy games* (Section 4.1.1) and then zoom in on *declining energy games* (Section 4.1.2), which include updates that combine vector components by taking their minimum. The latter are exactly what we need for the subsequent Section 4.2.

4.1.1 Monotonic Energy Games

To introduce energy games, we generalize the definitions on games (Definition 2.17 and the following) to include *energy levels*. We move beyond winning regions by defining quantitative winning *budgets*, which can still be characterized inductively.

Definition 4.1 (Energy reachability game). Given a partially ordered set of energies (\mathcal{E}, \leq) , an *energy reachability game* \mathcal{G}^\sharp is a reachability game $(G, G_d, \succrightarrow)$ extended by an edge labeling of energy updates $upd: (\succrightarrow) \rightarrow (\{\perp\} \cup \mathcal{E}) \rightarrow (\{\perp\} \cup \mathcal{E})$. We demand energy games to be *monotonic* in the following sense:

- $\perp \notin \mathcal{E}$ is final in the sense that $upd(m)(\perp) = \perp$ for $m \in \succrightarrow$.
- All update functions $u = upd(m)$ with $m \in \succrightarrow$ are monotonic and upward-closed with respect to \leq . More formally, this means that for any energies $e, e' \in \mathcal{E}$, if $e \in \text{dom}(u)$ and $e \leq e'$, then $e' \in \text{dom}(u)$ and $u(e) \leq u(e')$, where $\text{dom}(u) := \{e \in \mathcal{E} \mid u(e) \neq \perp\}$.

Definition 4.2 (Energy level as objective). For a finite play $\rho = g_0 g_1 \dots g_{n-1} \in G^*$ of \mathcal{G}^\sharp , starting from position g_0 with energy $e_0 \in \mathcal{E}$, the *energy level* $\text{EL}(\rho)$ is computed recursively:

- $\text{EL}(g_0) := e_0$
- $\text{EL}(g_0 \dots g_{i+1}) := \text{upd}(g_i, g_{i+1})(\text{EL}(g_0 \dots g_i))$

For infinite plays $\rho \in G^\omega$, we define energy levels to equal \perp .

We consider the attacker to be energy-bounded and understand $\text{EL}(\rho) = \perp$ to mean that they have run out of energy. Thus, we declare plays with $\text{EL}(\rho) = \perp$ to be won by the defender (even if they are stuck).

Strategies and winning strategies work exactly as in the energy-less scenario. Additionally, we lift positional strategies of Definition 2.19 to be *energy-positional* in the sense that they pick next moves depending on the current energy level, i.e. $s_a : G_a \times \mathcal{E} \rightarrow G$.

Generalizing the concept of winning regions as in Definition 2.20, we define *winning budgets* for game positions with overloaded notation:

Definition 4.3 (Winning budgets). For each position $g_0 \in G$ of energy game \mathcal{G}^\sharp , the *attacker winning budgets*, $\text{Win}_a^{\mathcal{G}^\sharp}(g_0) \subseteq \mathcal{E}$ are the energies e_0 where the attacker wins \mathcal{G}^\sharp from g_0 with e_0 . The defender winning budgets $\text{Win}_d^{\mathcal{G}^\sharp}$ are defined analogously.

In the context of a game, we write the shorthand Win_a^{\min} for *minimal attacker winning budgets*, $\text{Win}_a^{\min}(g_0) := \text{Min}(\text{Win}_a(g_0))$.

Classical energy games use \mathbb{N} -vectors for energies and \mathbb{Z} -vector addition for updates as in the following example:

Example 4.1 (A simple energy game). Consider vector energies $\mathcal{E} = \mathbb{N}^2$ with pointwise order and the energy game with graph as in Figure 4.2. Edges are labeled by update vectors $\vec{u} \in \mathbb{Z}^2$, each representing an update function

$$e \mapsto \begin{cases} e + \vec{u} & \text{if } e + \vec{u} \geq \mathbf{0} \\ \perp & \text{otherwise.} \end{cases}$$

How can the attacker win from g_1 ?

All attacker-won plays must end in g_6 to get the defender stuck.

For instance, the attacker cannot win from g_1 with $(0, 0)$. All outgoing paths would lead to \perp -energy and thus to the defender winning, before the game can reach g_6 .

But if the attacker starts with budget $(0, 2)$, they can take the upper path to g_4 with energy $(2, 1)$, from where both defender options lead to the defender being stuck in g_6 .

Also, if the attacker starts with $(2, 1)$, they win through the lower g_3 -path.

Moving to g_4 directly, would be more expensive than the g_2/g_3 -alternatives—so we can disregard this option in the monotonic context.

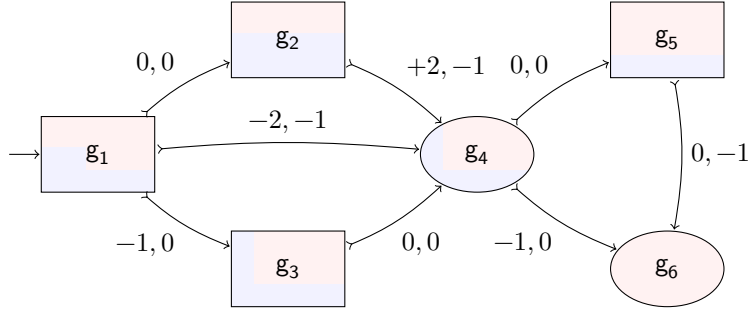


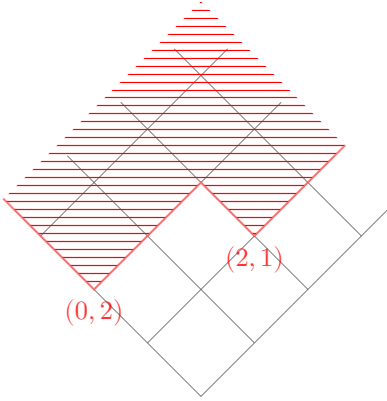
Figure 4.2: Simple energy game of Example 4.1.

Starting with less energy means that the defender has an option of bankrupting the attacker at g_4 (or before).

Summing up our observations, $(0, 2)$ and $(2, 1)$ define a Pareto front of minimal budgets where the attacker wins from g_1 , as depicted in Figure 4.3. The attacker and defender winning budgets of individual positions are also visualized by red and blue areas in Figure 4.2.

We can notice two things from the example:

1. A purely positional strategy would not suffice for the defender to win reliably at g_4 . Say the game starts with energy $(1, 1)$, that is, with a budget where the defender should win. Then, the level will either be $(0, 1)$ or $(3, 0)$ at g_4 . The defender has to either move to g_5 or g_6 , depending on the play so far. So, either the defender must know the history, or at least the current energy level to make an ideal decision.
2. Attacker's winning budgets are upward-closed, and defender budgets are downward-closed.

Figure 4.3: Attacker's Pareto front for g_1 of example energy game.

⁷² This is formalized as Lemma 3.2 in Lemke (2024).

⁷³ This is formalized as Corollary 3.1 in Lemke (2024).

Both are effects of Definition 4.1. More formally, the winning budgets are characterized by the following two propositions:

Proposition 4.1 (Upward closure of attacker budgets). *Given an energy game \mathcal{G} , the attacker winning budgets $\text{Win}_a^{\mathcal{G}}$ are pointwise upward-closed, that is, $\uparrow \text{Win}_a^{\mathcal{G}}(g) = \text{Win}_a^{\mathcal{G}}(g)$ for all positions $g \in G$.⁷²*

Proposition 4.2 (Winning budget determinacy). *Given an energy game \mathcal{G} , attacker budgets at a position g are attacker-winning if they are not defender-winning, and vice-versa.⁷³*

$$\text{Win}_a^{\mathcal{G}}(g) = \mathcal{E} \setminus \text{Win}_d^{\mathcal{G}}(g)$$

$$\text{Win}_d^{\mathcal{G}}(g) = \mathcal{E} \setminus \text{Win}_a^{\mathcal{G}}(g)$$

Proposition 4.3 (Inductive budgets). *Given an energy game \mathcal{G} , the attacker winning budgets $\text{Win}_a^{\mathcal{G}}$ are characterized inductively by the following rules:⁷⁴*

⁷⁴ Proved as Theorem 3 in Lemke (2024).

$$\frac{g_a \in G_a \quad g_a \rightsquigarrow g' \quad \text{upd}(g_a, g')(e) \in \text{Win}_a^G(g')}{e \in \text{Win}_a^G(g_a)}$$

$$\frac{g_d \in G_d \quad \forall u, g'. g_d \rightsquigarrow g' \longrightarrow \text{upd}(g_d, g')(e) \in \text{Win}_a^G(g')}{e \in \text{Win}_a^G(g_d)}$$

The inductive definition is nice as it allows for local proofs of how the attacker wins. Defender's wins on the other hand would dually be a *coinductive concept* because the defender wins loops.⁷⁵

The proofs in Lemke (2024) partially rely on the fact that one can derive reachability games from energy games, by inlining the energies and having the attacker stuck at exhausted energies.

Definition 4.4 (Derived reachability game). Given an energy game $\mathcal{G} = (G, G_d, \rightsquigarrow, \text{upd})$ over (\mathcal{E}, \leq) , the *derived reachability game* $\mathcal{G}^R = (G^R, G_d^R, \rightsquigarrow_R)$ is played on tuples $G^R := G \times (\mathcal{E} \cup \{\perp\})$ with $G_d^R := G_d \times \mathcal{E}$. Lifted moves $(g, e) \rightsquigarrow_R (g', e')$ are possible iff $e \neq \perp$, $g \rightsquigarrow g'$, and $e' = \text{upd}(g, g')(e)$.

Note the fine point that positions of exhausted energy that derive from defender positions are transformed to stuck *attacker* positions!

Winning regions of the reachability game and winning budgets of the energy game correspond.

Proposition 4.4 (Derived wins). $(g, e) \in \text{Win}_a^{\mathcal{G}^R}$ precisely if $e \in \text{Win}_a^{\mathcal{G}}(g)$.⁷⁶

⁷⁶ Lemma 3.4 in Lemke (2024).

Remark 4.1 (Generalized reachability). Ordinary reachability games can be seen as a special case of energy games with trivial energies, e.g. $\mathcal{E} = \{1\}$ and $\text{upd}(\cdot) = \text{id}_{\mathcal{E}}$. On such games, \mathcal{G} and \mathcal{G}^R according to Definition 4.4 have isomorphic game graphs.

Remark 4.2 (Our flavor of energy games). We deviate in two important points from other work on energy games:

- In the literature, it is more common to consider the defender instead of the attacker to be energy-bounded (e.g. Fahrenberg et al., 2011). This choice follows the intuition that there is a party with scarce resources that wants to keep the system running. For our purposes however, we want to bound the resources of an attacker, which is no fundamental change. Still, one cannot have both at the same time: Kupferman & Shamash Halevy (2022), studying both-bounded energy games, establish that their winner for a given energy is only decidable if both parties have only one-dimensional energies.
- We define energy-reachability games more abstractly than is common: In other publications, energy games work on numbers with component-wise updates as seen in Example 4.1. We do not demand this as, in the next sections, we will need updates that combine information from different components of the energy vector. Even prior generalizations on

⁷⁵ By the way, to those wondering: It is intentional that this thesis aims to achieve all its constructions without recurring to coinduction. Therefore, coinduction is only mentioned in side remarks and is not required to understand what is going on.

energy update functions such as Ésik et al. (2013) do not allow sufficient flexibility in this regard.

4.1.2 Declining Energy Games

We now turn to a special kind of energy games (introduced by Bisping, 2023b) with vector-valued energy levels where updates will never increase energy levels in any component. A d -dimensional *declining energy game* is played on energy vectors:

Definition 4.5 (Energies and energy updates). For dimensionality d , the set of *energies*, \mathbf{En} , is given by \mathbb{N}_{∞}^d . Energies are compared by \leq pointwise (cf. Example 3.2).

The set of *energy update labels*, \mathbf{Up} , contains $(u_1, \dots, u_d) \in \mathbf{Up}$ where each component u_k is a symbol of the form

- $u_k \in \{-1, 0\}$ (*relative update*), or
- $u_k = \min_D$ where $D \subseteq \{1, \dots, d\}$ and $k \in D$ (*minimum selection update*).

Applying an update to an energy, $\text{upd}(u, e)$, where $e = (e_1, \dots, e_d) \in \mathbf{En}$ and $u = (u_1, \dots, u_d) \in \mathbf{Up}$, yields a new energy vector e' where k th components are given by

- $e'_k := e_k + u_k$ for $u_k \in \{-1, 0\}$ and
- $e'_k := \min_{d \in D} e_d$ for $u_k = \min_D$.

Updates that would cause any component to become negative yield \perp .

Example 4.2 (Energy updates). Consider the update label $(\min_{\{1,2\}}, 0, -1)$.

- $\text{upd}((\min_{\{1,2\}}, 0, -1), (1, 1, 0))$ equals \perp because of the last component.
- $\text{upd}((\min_{\{1,2\}}, 0, -1), (2, 1, 1))$ equals $(1, 1, 0)$.
- $\text{upd}((\min_{\{1,2\}}, 0, -1), (1, 1, 1))$ equals $(1, 1, 0)$, as well.
- There is no $e \in \mathbf{En}$ such that $\text{upd}((\min_{\{1,2\}}, 0, -1), e) = (1, 0, 0)$, because the second component would demand a lower first component.

In summary, $\text{upd}((\min_{\{1,2\}}, 0, -1), \cdot)$ is neither injective nor surjective with respect to \mathbf{En} . The same is true for most other u and $\text{upd}(u, \cdot)$ with a min-update component.

Definition 4.6 (Declining energy game). Given a weight labeling $w: (\rightarrow) \rightarrow \mathbf{Up}$, a d -dimensional *declining energy game* $\mathcal{G}^{\blacktriangleright}$ is an energy reachability game with energies $\mathcal{E} := \mathbf{En}$ and with edges labeled $\text{upd}(m) := \text{upd}(w(m), \cdot)$.

We write $g \xrightarrow{u} g'$ for a move $g \rightarrow g'$ labeled by weight $w(g, g') = u$.

Declining energy games differ from more common energy games with vector addition updates as seen in Example 4.1 due to the non-invertibility of updates we encountered in Example 4.2.

We will use our new energy games to characterize not just single behavioral equivalences, but *whole spectra* of them.



Figure 4.4: The author, illustrating multi-dimensional energy updates on declining games to the audience of CAV'23. (Photo: Fei Bian)

4.2 Characterizing the P-easy Part of the Spectrum

As hinted at in Section 3.3.3, parts of the strong equivalence spectrum can be decided in polynomial time, whereas others require polynomial space (and thus effectively exponential time). Thanks to declining energy games, we can adapt the standard bisimulation game of Definition 2.22 to decide not just bisimulation, but *all polynomial-time-easy* equivalences of the strong spectrum *at once*.

4.2.1 The P-easy Slice

First, let us make explicit which equivalences we include in the polynomial-time slice of the spectrum. (The fact that they indeed can be decided efficiently is mostly well-known, but also itself a corollary of this section.)

To define the P-easy slice, we only use three dimensions:

1. modal depth,
2. depth of negative conjuncts, and
3. nesting depth of negations.

These are the first, fifth and sixth dimension of the strong spectrum in Definition 3.7. Components for other dimensions of the strong spectrum of the previous chapter are effectively set to ∞ .

Definition 4.7 (P-easy strong spectrum). We denote as *P-easy strong spectrum* the projection of the strong spectrum (Definition 3.7) to the first, fifth, and sixth dimension, thus by notions

$$\mathbb{N}^{\text{peasy}} := \mathbb{N}_{\infty}^3,$$

and the family of strong observation languages

$$\mathcal{O}_{(N_1, N_2, N_3) \in \mathbb{N}^{\text{peasy}}}^{\text{peasy}} := \mathcal{O}_{(N_1, \infty, \infty, \infty, N_2, N_3)}^{\text{strong}}.$$

In effect, $\mathcal{O}_{N \in \mathbb{N}^{\text{peasy}}}$ can be characterized by φ^N in the following grammar:

$$\begin{array}{ll} \varphi^N & ::= \top \\ & | \langle \alpha \rangle \varphi^{N - \hat{e}_1} \\ & | \bigwedge \{ \psi^N, \psi^N, \psi^N \dots \} \\ \psi^N & ::= \langle \alpha \rangle \varphi^{N - \hat{e}_1} \\ & | \neg \langle \alpha \rangle \varphi^{(N \sqcap (N_2, \infty, \infty)) - \hat{e}_1 - \hat{e}_3} \end{array}$$

Figure 4.5 gives names to coordinates that correspond to notions discussed previously.

Except for enabledness, E, all coordinates in Figure 4.5, after unfolding them for $\mathbb{N}^{\text{strong}}$, are identical to the ones used previously. When one

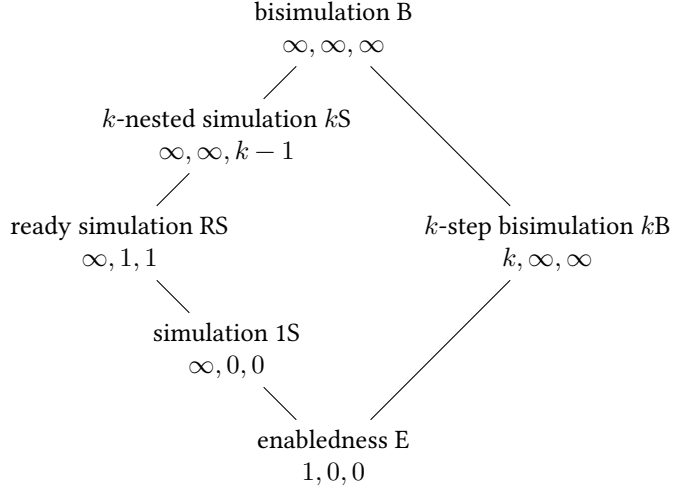


Figure 4.5: Hierarchy of polynomial-time decidable equivalences/preorders.

unfolds the coordinate for enabledness $(1, 0, 0)$ to the original strong system, $(1, \infty, \infty, \infty, 0, 0)$, it differs from the one previously used, namely, $(1, 0, 0, 0, 0, 0)$. The characterized language is strictly more expressive, as it contains conjunctions of enabled actions (e.g. $\bigwedge \{\langle a \rangle, \langle b \rangle\}$), but it is equally distinctive.⁷⁷

⁷⁷ lemma `Priced_Spectrum.lts.enabledness_conjunctions_are_neutral`

4.2.2 The Bisimulation *Energy* Game

Let us upgrade the bisimulation game \mathcal{G}_B of Definition 2.22 with energies. The claim is that the resulting game \mathcal{G}_B^\bullet will characterize all equivalences of the P-easy strong spectrum in the sense that $p \preceq_{\mathcal{O}_N} q$ for $N \in \mathbb{N}^{\text{peasy}}$ precisely if the defender wins \mathcal{G}_B^\bullet when the attacker starts from $[p, q]$ with budget $e = N$.

As we have seen in Section 2.4.5, the game moves match the distinctive power of productions in the $\mathcal{O}_{[B]}$ -grammar of Definition 2.16. Consequently, we can *count* the use of HML constructs in the game. As the game stands, we can meaningfully count the three dimensions used in Section 4.2.1.⁷⁸

We add the computations that happen in the grammar of Definition 4.7 as energy updates to the bisimulation game and obtain:

Definition 4.8 (Bisimulation energy game). For a transition system \mathcal{S} , the *bisimulation energy game* $\mathcal{G}_B^{\bullet\mathcal{S}}$ is played on the same graph as the *bisimulation game* $\mathcal{G}_B^{\mathcal{S}}$ (of Definition 2.22), but the moves are weighted by the following energy updates:

- Attacker simulation challenges count as an observation (using up the budget for modal depth):

$$[p, q] \xrightarrow{-1, 0, 0}_{\mathcal{B}}^\bullet (\alpha, p', q) \quad \text{if } p \xrightarrow{\alpha} p'.$$

⁷⁸ We cannot hope to count the other dimensions of Definition 3.7: The bisimulation game washes away the amount of *necessary* conjunctions as its HML has conjunctions under each observation, even though they might not be necessary for a distinguishing formula.

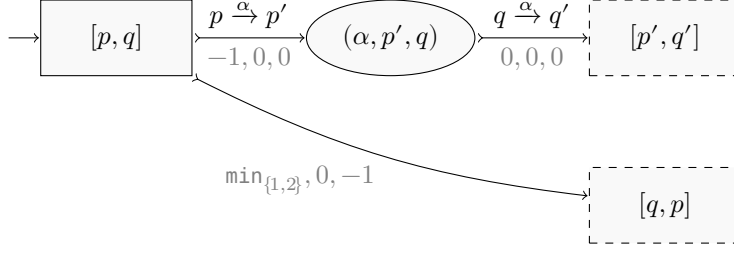


Figure 4.6: Game scheme of the bisimulation *energy* game with energy updates (Definition 4.8).

- Attacker swaps are counted as a negation and limit the further observations to the depth of negative conjuncts:

$$[p, q] \xrightarrow{\min_{\{1,2\}}, 0, -1}^{\text{B}} [q, p].$$

- And defender answers come for free:

$$(\alpha, p', q) \xrightarrow{0, 0, 0}^{\text{B}} [p', q'] \quad \text{if } q \xrightarrow{\alpha} q'.$$

Figure 4.6 visualizes the scheme of game rules, differing from Figure 2.13 only through energy updates.

The first dimension thus bounds how often the attacker may challenge simulation down the road, the third limits how often they may swap sides, and the middle dimension bounds the amount of simulation moves after a swap.

Example 4.3. Let us label the bisimulation game of Example 2.18 (distinguishing the “trolled philosophers”) with energy updates. Figure 4.7 shows the game graph, also exhibiting the cheapest formulas with regards to the polynomial spectrum that correspond to attacker winning strategies.

The attacker wins \mathcal{G}_B^{B} from $[Q, T]$ if they start out with an energy budget of $(2, 2, 2)$ or above. This space is visualized in Figure 4.8. But if the budget does not dominate this bound, the attacker loses. For instance, neither $(1, \infty, \infty)$ nor $(\infty, \infty, 1)$ is enough. The bound implies that the budgets, $(\infty, 1, 1)$ and $(\infty, 0, 0)$ are won by the defender as well.

On the game side, this tells us that the attacker needs at least two simulation moves (after swaps) and two swaps to tell Q apart from T . Telling apart T from Q is slightly easier, only requiring one swap (and only one simulation move after the swap).

But, due to the correspondence of the game to modal formulas and of the energy updates to the pricing of formulas in the spectrum, the attacker winning budgets tell us more: They reveal that one needs two observations and two (or more) negations to distinguish Q from T using formulas from $\mathcal{O}_{[B]}$.

Thus, $N_{Q,T}^{\text{peasy}} = \{N \in N^{\text{peasy}} \mid (2, 2, 2) \not\leq N\}$ is the solution for the (P-easy

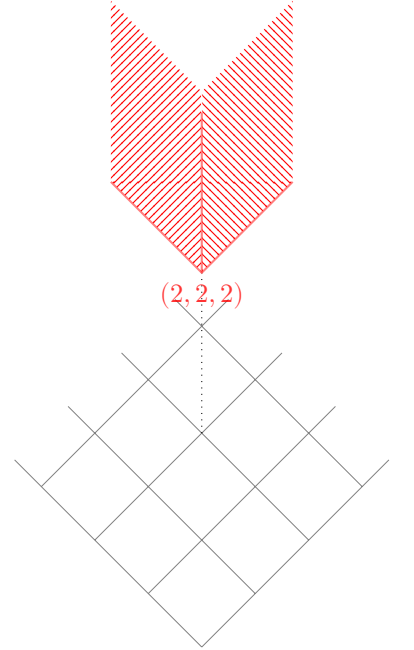


Figure 4.8: Attacker's 3D Pareto front for $[Q, T]$ of the example bisimulation energy game.

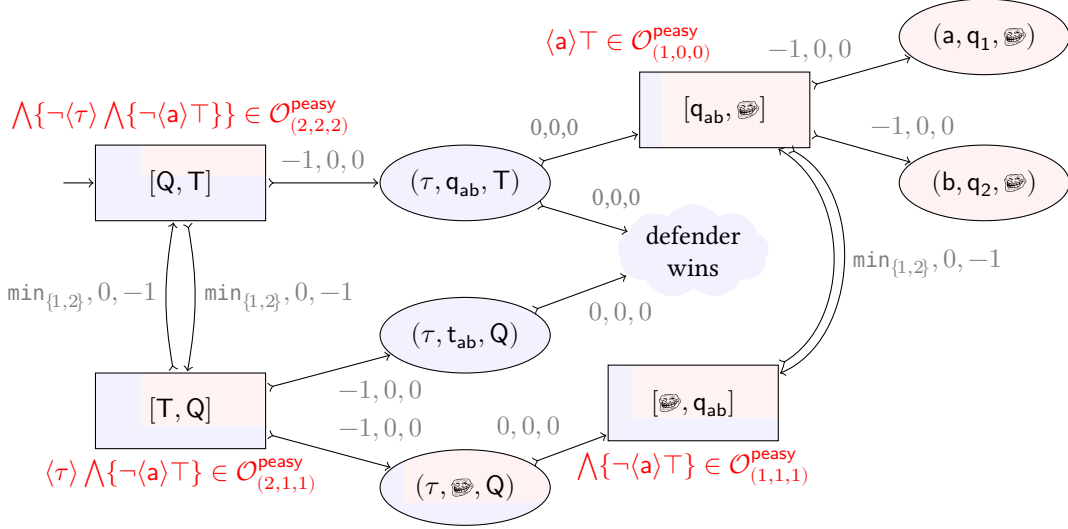


Figure 4.7: The bisimulation energy game of Example 4.3 with energy updates and implied formulas. Color of node backgrounds suggests the winning budgets of attacker (red) and defender (blue) in first and second dimension per position.

strong) spectroscopy problem. Q is preordered to T by 2-nested simulation, 1-step bisimulation, and all notions below; but distinguished by all above, in particular, 3-nested simulation and 2-step bisimulation.

This is in line with the solution we thought of for the original strong spectrum in Example 3.8.

4.2.3 Correctness of Characterization

We now prove that the bisimulation energy game indeed characterizes the equivalences of the P-easy strong spectrum. The section's proofs give the blueprint for proofs of stronger results for more general games in upcoming chapters.

The approach is to generalize the connection between winning attacks and distinguishing formulas in the bisimulation game. We have already explored this connection of strategies and logic in Lemma 2.9 and Lemma 2.10 and now extend it to energies and spectrum. One can say that the following is an “energized” version of Section 2.4.5. We show correspondence between notions and energies, $N \ni N = e \in \mathbf{En}$.

Definition 4.9 (Strategy formulas for $\mathcal{G}_B^{\mathbf{f}}$). The set of *strategy formulas* for a game position g and a budget e , $\text{Strat}_B(g, e)$, in the context of a bisimulation energy game $\mathcal{G}_B^{\mathbf{f}, S}$ is defined inductively by the rules in Figure 4.9.

Effectively, this definition generalizes the construction of distinguishing formulas on \mathcal{G}_B we have introduced in Lemma 2.9. Therefore, the formulas we encountered before also serve as examples of such strategy formulas.

$$\begin{array}{l}
\text{observation} \frac{[p, q] \xrightarrow{-1, 0, 0}_B^{\bullet} (\alpha, p', q) \quad e' = \text{upd}((-1, 0, 0), e) \in \text{Win}_a^{\mathcal{G}_B^{\bullet}}((\alpha, p', q)) \quad \varphi \in \text{Strat}_B((\alpha, p', q), e')}{(\langle \alpha \rangle \varphi) \in \text{Strat}_B([p, q], e)} \\
\\
\text{conjunction} \frac{\forall q' \in \text{Der}(q, \alpha). \quad (\alpha, p', q) \xrightarrow{0, 0, 0}_B^{\bullet} [p', q'] \wedge e \in \text{Win}_a^{\mathcal{G}_B^{\bullet}}([p', q']) \wedge \psi_{q'} \in \text{Strat}_B([p', q'], e)}{(\bigwedge_{q' \in \text{Der}(q, \alpha)} \psi_{q'}) \in \text{Strat}_B((\alpha, p', q), e)} \\
\\
\text{negation} \frac{[p, q] \xrightarrow{\min_{\{1, 2\}}, 0, -1}_B^{\bullet} [q, p] \quad e' = \text{upd}((\min_{\{1, 2\}}, 0, -1), e) \in \text{Win}_a^{\mathcal{G}_B^{\bullet}}([q, p]) \quad \langle \alpha \rangle \varphi \in \text{Strat}_B([q, p], e')}{(\neg \langle \alpha \rangle \varphi) \in \text{Strat}_B([p, q], e)}
\end{array}$$

Figure 4.9: Rules to derive cheap distinguishing formulas from attacker winning budgets.

The general design principle for the construction of strategy formulas as in Definition 4.9 is that each rule either corresponds to a kind of attacker move (observation/simulation, negation/swap) or a kind of defender position (conjunction).

Lemma 4.1 (Distinction soundness). *If $e \in \text{Win}_a^{\mathcal{G}_B^{\bullet}}([p, q])$, then there is $\varphi \in \text{Strat}_B([p, q], e)$ with $\varphi \in \mathcal{O}_e^{\text{peasy}}$, $p \in \llbracket \varphi \rrbracket$ and $q \notin \llbracket \varphi \rrbracket$.*

Proof. We prove the following (more general) property by induction.

1. If $e \in \text{Win}_a^{\mathcal{G}_B^{\bullet}}([p, q])$, then there is $\psi \in \text{Strat}_B([p, q], e)$ of form $\psi = \langle \alpha \rangle \varphi'$ or $\neg \langle \alpha \rangle \varphi'$ with $\bigwedge \{\psi\} \in \mathcal{O}_e^{\text{peasy}}$, $p \in \llbracket \psi \rrbracket$ and $q \notin \llbracket \psi \rrbracket$.
2. If $e \in \text{Win}_a^{\mathcal{G}_B^{\bullet}}((\alpha, p', q))$, then there is $\varphi \in \text{Strat}_B((\alpha, p', q), e)$ of form $\varphi = \bigwedge \Psi$ with $\varphi \in \mathcal{O}_e^{\text{peasy}}$, $p' \in \llbracket \varphi \rrbracket$ and $q \notin \llbracket \langle \alpha \rangle \varphi \rrbracket$.

Proof by induction on the inductive characterizations of attacker winning budgets (Proposition 4.3).

1. Assume $e \in \text{Win}_a^{\mathcal{G}_B^{\bullet}}([p, q])$. This must be due to one of the following moves:
 - Case $[p, q] \xrightarrow{\bullet}_B [q, p]$
with $e' = \text{upd}((\min_{\{1, 2\}}, 0, -1), e) \in \text{Win}_a([q, p])$. By induction hypothesis on $[q, p]$, there is $\psi \in \text{Strat}_B([q, p], e')$ with $\bigwedge \{\psi\} \in \mathcal{O}_{e'}^{\text{peasy}}$, $q \in \llbracket \psi \rrbracket$ and $p \notin \llbracket \psi \rrbracket$. Consider the possible forms of ψ :
 - $\psi = \langle \alpha \rangle \varphi'$. By Definition 4.9 of Strat_B , we obtain $\neg \langle \alpha \rangle \varphi' \in \text{Strat}_B([p, q], e)$. Because of the semantics of HML, $\neg \langle \alpha \rangle \varphi'$ must distinguish p from q . Also, $\bigwedge \{\neg \langle \alpha \rangle \varphi'\} \in \mathcal{O}_e^{\text{peasy}}$, by the calculations of the grammar in Definition 4.7.
 - $\psi = \neg \langle \alpha \rangle \varphi'$. This can only be in $\text{Strat}_B([q, p], e')$ due to $\langle \alpha \rangle \varphi'$ being in $\text{Strat}_B([p, q], \text{upd}((\min_{\{1, 2\}}, 0, -1), e'))$. Clearly, $\langle \alpha \rangle \varphi'$ must distinguish p from q and $\bigwedge \{\langle \alpha \rangle \varphi'\} \in \mathcal{O}_e^{\text{peasy}}$.
 - Case $[p, q] \xrightarrow{\bullet}_B (\alpha, p', q)$
with $p \xrightarrow{\alpha} p'$ and $e' = \text{upd}((-1, 0, 0), e) \in \text{Win}_a((\alpha, p', q))$.

By induction hypothesis on (α, p', q) , there is $\bigwedge \Psi \in \text{Strat}_B((\alpha, p', q), e')$ and $\bigwedge \Psi \in \mathcal{O}_{e'}^{\text{peasy}}$ with $p' \in \llbracket \bigwedge \Psi \rrbracket$ and $q \notin \llbracket \langle \alpha \rangle \bigwedge \Psi \rrbracket$. By Definition 4.9, we obtain $\langle \alpha \rangle \bigwedge \Psi \in \text{Strat}_B([p, q], e)$. Due to the semantics of HML and $p \xrightarrow{\alpha} p'$, $p \in \llbracket \langle \alpha \rangle \bigwedge \Psi \rrbracket$, thus distinguishing p from q . Also, $\langle \alpha \rangle \bigwedge \Psi \in \mathcal{O}_e^{\text{peasy}}$ as $\bigwedge \Psi \in \mathcal{O}_{e-\hat{e}_1}^{\text{peasy}}$.

2. Assume $e \in \text{Win}_a^{\mathcal{G}_B^*}((\alpha, p', q))$. This means that e suffices for the attacker to win every move $(\alpha, p', q) \xrightarrow{\bullet}_B [p', q']$ with $q \xrightarrow{\alpha} q'$ that the defender might take, with $e \in \text{Win}_a([p', q'])$. For each q' , we employ the induction hypothesis to obtain $\psi_{q'} \in \text{Strat}_B([p', q'], e)$ of form $\psi_{q'} = \langle \alpha \rangle \varphi'$ or $\neg \langle \alpha \rangle \varphi'$ with $\bigwedge \{\psi_{q'}\} \in \mathcal{O}_e^{\text{peasy}}$, $p' \in \llbracket \psi_{q'} \rrbracket$ and $q' \notin \llbracket \psi_{q'} \rrbracket$. By Definition 4.9, $(\bigwedge_{q' \in \text{Der}(q, \alpha)} \psi_{q'}) \in \text{Strat}_B((\alpha, p', q), e)$. By the HML semantics, $(\bigwedge_{q' \in \text{Der}(q, \alpha)} \psi_{q'})$ must be true for p' , and also taking the semantics of observation, $\langle \alpha \rangle (\bigwedge_{q' \in \text{Der}(q, \alpha)} \psi_{q'})$ false for q . Moreover, $(\bigwedge_{q' \in \text{Der}(q, \alpha)} \psi_{q'}) \in \mathcal{O}_e^{\text{peasy}}$ by the grammar. \square

We have thus established how to connect from energies to notions through formulas. Now let us link back.

Lemma 4.2 (Distinction completeness). *If there is $\varphi \in \mathcal{O}_N^{\text{peasy}}$ with $p \in \llbracket \varphi \rrbracket$ and $q \notin \llbracket \varphi \rrbracket$, then $N \in \text{Win}_a^{\mathcal{G}_B^*}([p, q])$.*

Proof. By induction on the grammar of $\mathcal{O}_N^{\text{peasy}}$.

Consider the cases of $\varphi \in \mathcal{O}_N^{\text{peasy}}$ with $p \in \llbracket \varphi \rrbracket$ and $q \notin \llbracket \varphi \rrbracket$.

- $\varphi = \top$. This cannot be the case as $q \notin \llbracket \varphi \rrbracket$ and $\llbracket \top \rrbracket = \mathcal{P}$.
- $\varphi = \langle \alpha \rangle \varphi'$ with $\varphi' \in \mathcal{O}_{N-\hat{e}_1}^{\text{peasy}}$. As φ distinguishes p from q , there must be a $p' \in \llbracket \varphi' \rrbracket$ such that $p \xrightarrow{\alpha} p'$ and, for all $q' \in \text{Der}(q, \alpha)$, $q' \notin \llbracket \varphi' \rrbracket$. Due to the induction hypothesis, $N - \hat{e}_1 \in \text{Win}_a([p', q'])$ for all such $q' \in \text{Der}(q, \alpha)$. Therefore, $N - \hat{e}_1 \in \text{Win}_a((\alpha, p', q))$. As the attacker can move $[p, q] \xrightarrow{-\hat{e}_1} (\alpha, p', q)$, this proves $N \in \text{Win}_a([p, q])$ by Proposition 4.3.
- $\varphi = \bigwedge \Psi$ one of the $\psi \in \Psi$ must be false for q . As the grammar does not affect N for ψ , we know $\bigwedge \{\psi\} \in \mathcal{O}_N^{\text{peasy}}$. Consider the possible forms of ψ :
 - Case $\psi = \langle \alpha \rangle \varphi'$ with $\varphi' \in \mathcal{O}_{N-\hat{e}_1}^{\text{peasy}}$. Then we can apply the same argument as in the case of $\varphi = \langle \alpha \rangle \varphi'$ to infer that $N \in \text{Win}_a([p, q])$.
 - Case $\psi = \neg \langle \alpha \rangle \varphi'$ with $\varphi' \in \mathcal{O}_{(N \sqcap (N_2, \infty, \infty)) - \hat{e}_1 - \hat{e}_3}^{\text{peasy}}$. Therefore, $\langle \alpha \rangle \varphi' \in \mathcal{O}_{(N \sqcap (N_2, \infty, \infty)) - \hat{e}_3}^{\text{peasy}}$ distinguishes q from p . This time, we can employ the same argument as in the case of $\varphi = \langle \alpha \rangle \varphi'$ to obtain that $(N \sqcap (N_2, \infty, \infty)) - \hat{e}_3 \in \text{Win}_a([q, p])$. The move $[p, q] \xrightarrow{\min\{1, 2\}, 0, -1}_B [q, p]$ and $\text{upd}((\min\{1, 2\}, 0, -1), N) = (N \sqcap (N_2, \infty, \infty)) - \hat{e}_3$ justify that $N \in \text{Win}_a([p, q])$. \square

The crucial trick here has been that one does not need all conjuncts of a conjunction to establish a distinction and that focusing on parts can be done without leaving \mathcal{O}_N .

Combining Lemma 4.1 with Lemma 4.2, and taking the defender perspective, we immediately get:

Theorem 4.1 ($\mathbb{N}^{\text{peasy}}$ -characterization). $p \preceq_{\mathcal{O}_N} q$ for $N \in \mathbb{N}^{\text{peasy}}$ precisely if the defender wins \mathcal{G}_B^\bullet from $[p, q]$ with budget N .

Thus, we have established that the bisimulation energy game \mathcal{G}_B^\bullet characterizes the equivalences of the P-easy strong spectrum $\mathbb{N}^{\text{peasy}}$. This generalizes Theorem 2.3 through energy games.

To exploit this property to decide equivalences and solve the spectroscopy problem algorithmically, we need one more ingredient: A decision procedure for winning budgets in declining energy games.

4.3 Deciding Energy Games

This section is about how to compute what energy budgets are winning for the attacker (or the defender) at positions of an energy game:

i Problem 2: Energy game winner problem

In the context of an energy system (\mathcal{E}, \leq) and a finite monotonic energy game $\mathcal{G}^\bullet = (G, G_a, \rightarrow, upd)$, the *energy game winner problem* goes:

Input Game position $g \in G$.

Output $\text{Win}_a^{\min}(g)$ —the Pareto front of minimal winning budgets for the attacker at g .

We will show how to solve the problem for energy games with updates that can be undone in a certain sense. To this end, we will adapt the idea of back-propagating how the defender loses in reachability games from Algorithm 2.1.

The change is that we have to propagate not just attacker wins, but Pareto fronts of attacker-winning energy levels. For this, we have to “invert” the energy update functions. But, as we have seen in Example 4.2, our specific declining energy updates are neither injective nor surjective, and thus cannot be cleanly inverted.

We will tackle these challenges using *Galois connections*. For details, readers are referred to Lemke (2024).

4.3.1 Galois Connections

Galois connections can be thought of as a pair of *monotonic functions that invert each other up to their partial ordering relations*. Another standard intuition

is that Galois connections couple a *concrete* and a more *abstract* domain. This pairing is what the variable naming in the following definition alludes to.

Definition 4.10 (Galois connection). A *Galois connection* between two partially ordered sets (C, \leq_C) and (A, \leq_A) is a pair of functions $\alpha: C \rightarrow A$ and $\gamma: A \rightarrow C$ such that, for all $a \in A$ and $c \in C$:

$$\alpha(c) \leq_A a \iff c \leq_C \gamma(a).$$

An illustration of the connection property can be seen in Figure 4.10.

Example 4.4 (\mathbb{N} as abstraction for \mathbb{R}). Intervals of positive real numbers can be abstracted into natural numbers, as illustrated by Figure 4.11. More formally:

Adopt the non-negative reals $\mathbb{R}^{\geq 0}$ as concrete domain and the natural numbers \mathbb{N} as abstract domain. Then, as function $\alpha_{\mathbb{R}}: \mathbb{R}^{\geq 0} \rightarrow \mathbb{N}$ take flooring, $x \mapsto \lfloor x \rfloor$, and for the other direction, the identity $\text{id}_{\mathbb{N}}, n \mapsto n$. Clearly, $\lfloor x \rfloor \leq n \iff x \leq n$. Therefore, $\alpha_{\mathbb{R}}$ and $\text{id}_{\mathbb{N}}$ form a Galois connection.

Many monotonic functions naturally induce a Galois-connected abstraction function through min.

Lemma 4.3. *The following are equivalent:*

- $\gamma: A \rightarrow C$ is a monotonic function, and $\alpha(c) = \min\{a \mid c \leq_C \gamma(a)\}$ for all $c \in C$.
- α and γ constitute a Galois connection between C and A .⁷⁹

We will refer to functions derived like α in Lemma 4.3 as “undo” functions. We understand them to generalize inverse functions of those monotonic functions where $\min\{a \mid c \leq_C \gamma(a)\}$ is defined for any c . This intuition can also be expressed more formally as the following proposition:

Proposition 4.5 (Inversion as Galois connection). Assume a function $f: A \rightarrow C$ has a unique inverse function $f^{-1}: C \rightarrow A$ such that $f^{-1} \circ f = \text{id}_A$ and $f \circ f^{-1} = \text{id}_C$.

If f is monotonic with respect to some partial orders on C and A , then f^{-1} and f must form a Galois connection.

Proof. Because of injectivity and monotonicity, f and f^{-1} , must be (strictly) monotonic. Together with the definition of function inversion, we can reason

$$f^{-1}(c) \leq_A a \implies f(f^{-1}(c)) \leq_A f(a) \implies c \leq_C f(a)$$

and

$$c \leq_C f(a) \implies f^{-1}(c) \leq_A f^{-1}(f(a)) \implies f^{-1}(c) \leq_A a. \quad \square$$

In the following, we will use a \mathcal{U} -symbol to label the functions that we use to undo energy updates.

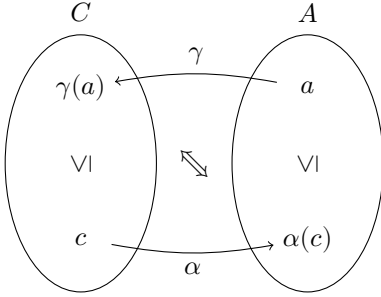


Figure 4.10: Visualization of Galois connections according to Definition 4.10.

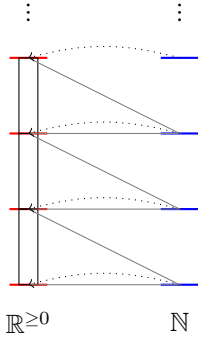


Figure 4.11: Illustration of the Galois connection of Example 4.4 between $\mathbb{R}^{\geq 0}$ and \mathbb{N} using $\lfloor \cdot \rfloor$ -function, whose mapping is illustrated by the gray triangles.

⁷⁹ Proposition 4 of Ern  et al. (1993).

```

1  def compute_winning_budgets( $\mathcal{G}^\sharp = (G, G_d, \succrightarrow, upd), upd^\sharp$ ):
2      attacker_win :=  $[g \mapsto \emptyset \mid g \in G]$ 
3      todo :=  $\{g \in G_d \mid g \not\succrightarrow\}$ 
4      while todo  $\neq \emptyset$ :
5          g := some todo
6          todo := todo  $\setminus \{g\}$ 
7          if  $g \in G_a$ :
8              new_attacker_win :=  $\text{Min}(\{upd^\sharp(g, g')(e') \mid$ 
9                   $g \succrightarrow g' \wedge e' \in \text{attacker\_win}[g']\})$ 
10             else :
11                 new_attacker_win :=  $\{0\}$ 
12                 for  $g' \in (g \succrightarrow \cdot)$ :
13                     new_attacker_win :=  $\text{Min}(\{\sup\{e_a, upd^\sharp(g, g')(e')\} \mid$ 
14                          $e_a \in \text{new\_attacker\_win} \wedge e' \in \text{attacker\_win}[g']\})$ 
15                 if new_attacker_win  $\neq \text{attacker\_win}[g]$ :
16                     attacker_win[g] := new_attacker_win
17                     todo := todo  $\cup (\cdot \succrightarrow g)$ 
18  Winamin := attacker_win
19  return Winamin

```

Algorithm 4.1: Algorithm determining the minimal attacker winning budgets Win_a^{\min} of an energy game \mathcal{G}^\sharp with undo functions upd^\sharp .

Remark 4.3 (Galois connections on the spectrum). Definition 3.5 demands monotonicity of our observation languages for a spectrum, $\mathcal{O}_{N \in \mathbb{N}}$. If we construct an abstraction function as in Lemma 4.3 to undo the language selection for a set of formulas Φ , the result would read:

$$\alpha_{\mathcal{O}}(\Phi) := \min\{N \in \mathbb{N} \mid \Phi \subseteq \mathcal{O}_N\}.$$

If this function is defined, then its singleton case exactly matches our expressiveness prices of Definition 3.6:

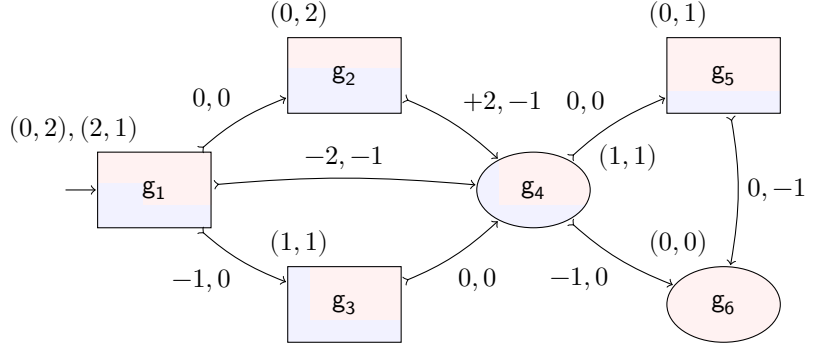
$$\text{expr}(\varphi) := \min\{N \in \mathbb{N} \mid \varphi \in \mathcal{O}_N\}.$$

4.3.2 The Algorithm

To implement a *backpropagation* algorithm, we have to assume that we know a function that undoes energy updates, $upd^\sharp: (\succrightarrow) \rightarrow \mathcal{E} \rightarrow \mathcal{E}$ with $upd^\sharp(m)(e') = \min\{e \mid e' \leq upd(m)(e)\}$. That means, we assume that there is a Galois connection $(upd^\sharp(m), upd(m))$ between energies \mathcal{E} and the domain of updates $\{e \in \mathcal{E} \mid upd(m)(e) \neq \perp\}$ for each $m \in \succrightarrow$.

Following Lemke (2024), we assume that the energy system (\mathcal{E}, \leq) is a well-founded bounded sup-semi-lattice. By 0 , we denote its minimal element.

Algorithm 4.1 computes the minimal attacker winning budgets.

Figure 4.12: Computed winning budgets Win_a^{\min} on the game in Example 4.5.

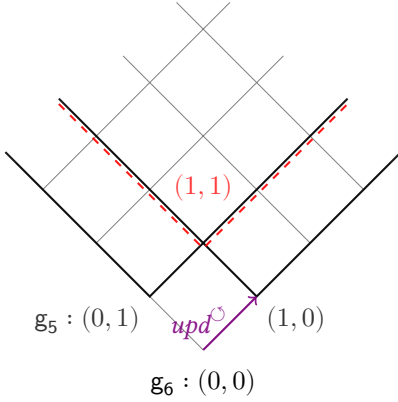
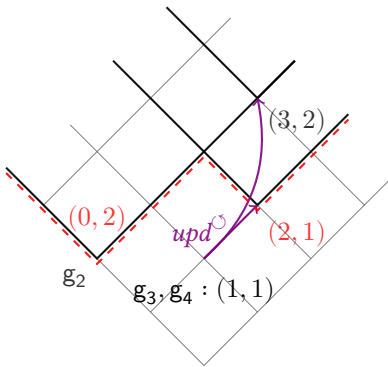
The algorithm can be understood as a generalization of Algorithm 2.1, but as positions might need to be revisited more than once, we have to depart from the option counting trick. Whenever we learn about a new minimal energy for the attacker to win, we schedule a position to be (re-)visited. The chain reaction starts at defender positions with no outgoing moves (lines 3 and 10).

We illustrate the algorithm by running it on the simple energy game of Example 4.1.

Example 4.5 (Deciding the simple energy game). Let us execute Algorithm 4.1 on the simple \mathbb{N}^2 -game of Example 4.1. The game graph, labeled by the minimal attacker winning energies we find, is reproduced in Figure 4.12.

What we need for the algorithm to run is some undo function $\text{upd}^{\mathcal{J}} : (\succ) \rightarrow \mathcal{E} \rightarrow \mathcal{E}$. We define it as:

$$\text{upd}^{\mathcal{J}}(m)(e') := \sup\{e - \vec{u}, 0\} \text{ if } m \text{ is } \vec{u}\text{-labeled}$$

Figure 4.13: Combination (dashed red) of successor Pareto fronts (black) to intersect at defender position g_4 in step 4.Figure 4.14: Combination (dashed red) of successor Pareto fronts (black) to take the union at attacker position g_1 in step 8.

We list the steps of the computation in Table 4.1, and explain the mechanics below.

0. The algorithm begins with g_6 in the todo-set as this is the only position where the defender is stuck.
1. Updating the defender node $g_6 \in G_d$, we set its minimal attacker-winning budget to 0. This means that the attacker wins here with any budget. As there has been a change, both predecessors g_4 and g_5 are added to the todo. (Their order is undefined; for the sake of the simulation we will always pick the first positions from the todo.)
2. Updating $g_4 \in G_d$ finds no attacker wins so far because we do not yet know a way for the attacker to win g_5 , that is, $\text{attacker_win}[g_5] = \emptyset$.
3. $g_5 \in G_a$ is updated with the $\text{upd}^{\mathcal{J}}((0, -1))((0, 0)) = \sup\{(0, 0) - (0, -1), 0\} = (0, 1)$. This places g_4 in todo again.
4. Now, we can find attacker winning budgets for $g_4 \in G_d$: The two successor options (after undo-update) are $(0, 1)$ and $(1, 0)$, as depicted in Figure 4.13. Unless the attacker at least arrives with energy

$\sup\{(0, 1), (1, 0)\} = (1, 1)$, the defender has a way of winning. Effectively, this amounts to intersecting attacker winning budgets of possible successor positions.

5. For $g_1 \in G_a$, it suffices to know a winning budget for one successor. Therefore, we can determine as a preliminary winning budget $upd^{\mathcal{S}}((-2, -1))((1, 1)) = (3, 2)$.
6. $g_2 \in G_a$ also derives its winning budget from $(1, 1)$. But $(1, 1) - (2, -1) = (-1, 2)$ would be outside the possible energy levels the attacker could carry here and is thus sup-ed into \mathbb{N}^2 , yielding new win $(0, 2)$.
7. $g_3 \in G_a$ just propagates $(1, 1)$ from g_4 .
8. Revisiting $g_1 \in G_a$, we can now combine the three Pareto-fronts of its successors as illustrated in Figure 4.14 as $\text{Min}\{(0, 2), (2, 1), (3, 4)\} = \{(0, 2), (2, 1)\}$. This operation corresponds to taking the union of the three sets of attacker winning budgets. The previous minimal budget $(3, 4)$ of step 5 is thus superseded. As *todo* is empty, the algorithm terminates.

Theorem 4.2 (Correctness). *Assume energies form a well-founded bounded sup-semi-lattice (\mathcal{E}, \leq) . Given a finite-state energy game $\mathcal{G}^\sharp = (G, G_d, \mapsto, upd)$ with computable undo function $upd^{\mathcal{S}}(m)(e') = \min\{e \mid e' \leq upd(m)(e)\}$ for all moves, Algorithm 4.1 computes the Pareto front of minimal attacker winning budgets, solving the energy game winner problem (Problem 2).*

Proof. Algorithm 4.1 is a work-list variant of a fixed-point iteration computing the least fixed point of attacker winning budgets Win_a according to the inductive characterization of Proposition 4.3.

The “least” here refers to the size of the Win_a -sets characterized by the Win_a^{\min} -Pareto front.

A detailed proof, employing Kleene’s fixed point theorem, can be found in

Table 4.1: Steps while solving the game in Example 4.5.

step	g	new_attacker_win	todo
0	–	$g_i \mapsto \emptyset$	g_6
1	g_6	$\{(0, 0)\}$	g_4, g_5
2	g_4	\emptyset	g_5
3	g_5	$\{(0, 1)\}$	g_4
4	g_4	$\{(1, 1)\}$	g_1, g_2, g_3
5	g_1	$\{(3, 2)\}$	g_2, g_3
6	g_2	$\{(0, 2)\}$	g_3, g_1
7	g_3	$\{(1, 1)\}$	g_1
8	g_1	$\{(0, 2), (2, 1)\}$	\emptyset

Lemke (2024). Theorem 5 of Lemke (2024) refers to a variant of Algorithm 4.1, where *all* game positions are updated *simultaneously* in each iteration, which aligns more directly with the underlying functor. As is common for graph algorithms, Algorithm 4.1 with updates of single vertices whose neighborhood has changed computes the same result and, usually, in a more efficient manner. \square

4.3.3 Complexity

Thanks to Lemke (2024), we can name quite precise bounds for the running time of the algorithm in terms of size of the game and shape of the energy lattice.

Definition 4.11. The *width*, $\text{wdh}_B(b)$, of a (semi-)lattice (B, \leq) below a point b is defined as the maximal size of anti-chains $B' \subseteq \downarrow \{b\}$. (Cf. Definition 3.4.)

Theorem 4.3 (Complexity). Consider the following parameters of an energy game problem on $\mathcal{G}^\sharp = (G, G_d, \succrightarrow, \text{upd})$ where the energies at least form a well-founded bounded sup-semi-lattice (\mathcal{E}, \leq) :

- t_{sup} , time to calculate the supremum between two elements of \mathcal{E} ,
- t_{\leq} , time to compare two elements of \mathcal{E} ,
- t_{upd^\sharp} , time to compute $\text{upd}^\sharp(\cdot)$ on an energy,
- $e_{\mathcal{G}}$, the highest energy that can be achieved by applying permutations of $\text{upd}^\sharp(\cdot)(\cdot)$ to $\mathbf{0}$ for $|G| - 1$ times.
- o_{\succrightarrow} , the out-degree of the game graph \succrightarrow .

Then, Algorithm 4.1 computes the minimal attacker winning budgets in

$$O((|G|^2 + |\downarrow \{e_{\mathcal{G}}\}|) \cdot |G| \cdot o_{\succrightarrow} \cdot (\text{wdh}_{\mathcal{E}}(e_{\mathcal{G}}))^2 \cdot (t_{\text{upd}^\sharp} + t_{\text{sup}} + \text{wdh}_{\mathcal{E}}(e_{\mathcal{G}}) \cdot t_{\leq})).$$

If upd is declining, then the following suffices:

$$O(|G|^2 \cdot o_{\succrightarrow} \cdot (\text{wdh}_{\mathcal{E}}(e_{\mathcal{G}}))^2 \cdot (t_{\text{upd}^\sharp} + t_{\text{sup}} + \text{wdh}_{\mathcal{E}}(e_{\mathcal{G}}) \cdot t_{\leq})).$$

In both cases, the algorithm needs $O(|G| \cdot \text{wdh}_{\mathcal{E}}(e_{\mathcal{G}}) \cdot s_{\mathcal{E}})$ space to store the Pareto fronts, where $s_{\mathcal{E}}$ bounds the space needed per energy. For declining energy games, it is reasonable to assume $s_{\mathcal{E}} \in O(d)$.⁸⁰

4.3.4 Solving Declining Energy Games

The preceding two subsections have established facts that we now only need to instantiate to declining energy games. For this, we must instantiate upd^\sharp to obtain an algorithm and think about the structure of the energies to tell its complexity.

Definition 4.12 (Undoing declining updates). The undo-update function upd^\sharp on a d -dimensional declining energy game is defined component-wise for $k \in$

⁸⁰ As in Theorem 4.2, we refer to Lemke (2024) for a very detailed proof, where the corresponding fact is proved as Theorem 7.

$\{1, \dots, d\}$ as:

$$(\text{upd}^\circ(u, e'))_k := \max(\{e'_k - u_k \mid u_k \in \{0, -1\}\} \cup \{e'_k \mid \exists D. u_j = \min_D \wedge k \in D\}).$$

Example 4.6. Consider the update label $(\min_{\{1,2\}}, -1)$ (a 2D-sibling of the one from Example 4.2). Its undo-update according to Definition 4.12 computes $\text{upd}^\circ((\min_{\{1,2\}}, -1), (e_1, e_2)) = (\max(e_1, e_2), e_2 + 1)$.

Figure 4.15 shows how updates and undo-updates transform 2D-coordinates.

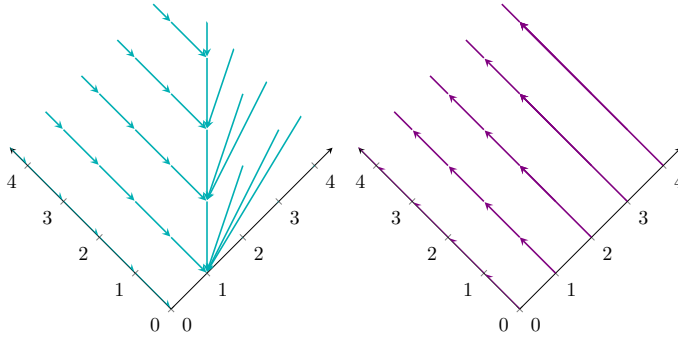


Figure 4.15: Effects of updates $\text{upd}((\min_{\{1,2\}}, -1), \cdot)$ and their undoing via $\text{upd}^\circ((\min_{\{1,2\}}, -1), \cdot)$.

For instance, we can observe the handling of non-invertibility:

- $\text{upd}^\circ((\min_{\{1,2\}}, -1), (2, 1)) = (2, 2)$
- $\text{upd}^\circ((\min_{\{1,2\}}, -1), (2, 0)) = (2, 2)$
- $\text{upd}((\min_{\{1,2\}}, -1), (4, 2)) = (2, 1)$
- $\text{upd}((\min_{\{1,2\}}, -1), (2, 2)) = (2, 1)$

Intuitively, updates including $\min_{\{...\}}$ cap coordinates; the undo-update then rounds into the space from where a coordinate would safely be dominated.

Lemma 4.4. For all updates $u \in \text{Up}$, $\text{upd}^\circ(u, \cdot)$ and $\text{upd}(u, \cdot)$ form a Galois connection on the domain $\text{dom}(\text{upd}(u, \cdot))$.

Proof. As $\text{upd}(u, \cdot)$ is monotonic, all that needs to be shown according to Lemma 4.3, is that $\text{upd}^\circ(u, e') = \min\{e \mid e' \leq \text{upd}(u, e)\}$.

- Soundness: $\text{upd}^\circ(u, e') \in \{e \mid e' \leq \text{upd}(u, e)\}$. This boils down to showing $e'_k \leq (\text{upd}(u, \text{upd}^\circ(u, e')))_k$. Consider the cases of u_k .
 - Case $u_k \in \{0, -1\}$. Then, we have to show $e'_k \leq (\text{upd}^\circ(u, e'))_k + u_k$. Due to the max-definition of upd° , this must be the case as $e'_k = e'_k - u_k + u_k \leq (\text{upd}^\circ(u, e'))_k + u_k$.

- Case there is D such that $u_k = \min_D$ and $k \in D$. Then, by definition of min-updates, we have to show $e'_k \leq \min\{\text{upd}^\circ(u, e')_j \mid j \in D\}$, that is, $e'_k \leq \text{upd}^\circ(u, e')_j$ for all $j \in D$. We do so by contradiction: Assume there were a j where this ordering does not hold, meaning $e'_k > \text{upd}^\circ(u, e')_j$. As $k \in D$, this is impossible due to the definition of upd° in terms of max.
- Minimality: For all $e' \leq \text{upd}(u, e)$, we show $\text{upd}^\circ(u, e') \leq e$. More specifically, $(\text{upd}^\circ(u, e'))_k \leq e_k$. Let us define $ma := \max(\{0\} \cup \{e'_j \mid \exists D. u_j = \min_D \wedge k \in D\})$. Consider the cases of $u_k \in \text{Up}$.
 - Case $u_k \in \{0, -1\}$ and $e'_k - u_k \geq ma$. Then $e'_k \leq (\text{upd}(u, e))_k = e_k + u_k$, implying $e'_k - u_k \leq (\text{upd}(u, e))_k = e_k$. By definition, $(\text{upd}^\circ(u, e'))_k = \max(e'_k - u_k, ma)$. As $e'_k - u_k \geq ma$, we can connect to the previous inequality, getting $(\text{upd}^\circ(u, e'))_k = e'_k - u_k \leq e_k$.
 - Otherwise, $(\text{upd}^\circ(u, e'))_k = ma$. We show $ma \leq e_k$ by contradiction. Assume it were the case that $ma > e_k$. For this to be true there would need to be j and D such that $u_j = \min_D$, $k \in D$, and $e'_j \geq ma > e_k$. By definition of min-update at j , we calculate $(\text{upd}(u, e))_j \leq e_k < e'_j$. But this would contradict the global assumption that $e' \leq \text{upd}(u, e)$. \square

Lemma 4.5 (Declining complexity). *Algorithm 4.1 solves the energy game problem on a d -dimensional declining energy game $\mathcal{G}^\blacktriangleright = (G, G_d, \multimap, w)$ in time*

$$O(o_{\multimap} \cdot |G|^{2 \cdot d} \cdot (d^2 + |G|^{d-1} \cdot d))$$

and space $O(|G|^d \cdot d)$.

Proof. For declining updates, we instantiate the second case of Theorem 4.3:

$$O(|G|^2 \cdot o_{\multimap} \cdot (\text{wdh}_{\mathcal{E}}(e_{\mathcal{G}}))^2 \cdot (t_{\text{upd}^\circ} + t_{\text{sup}} + \text{wdh}_{\mathcal{E}}(e_{\mathcal{G}}) \cdot t_{\leq})).$$

For *declining energy games* as defined in Definition 4.6, we can fill in several blanks:

- t_{sup} , the time to calculate the supremum between two elements of En is in $O(d)$,
- t_{\leq} , time to compare two elements of En is in $O(d)$,
- t_{upd° , time to compute upd° on an energy is in $O(d^2)$,
- $e_{\mathcal{G}}$, the highest energy that can be achieved by applying permutations of upd° to $\mathbf{0}$ for $|G| - 1$ times, is the d -ary vector $(|G| - 1, \dots, |G| - 1)$,
- $\text{wdh}_{\text{En}}(e_{\mathcal{G}})$, the widest anti-chain under $e_{\mathcal{G}}$, can be bounded $O(|G|^{d-1})$.

This yields:

$$\begin{aligned}
& O(|G|^2 \cdot o_{\rightarrow} \cdot (\text{wdh}_{\text{En}}(e_{\mathcal{G}}))^2 \cdot (t_{\text{upd}^{\mathcal{S}}} + t_{\text{sup}} + \text{wdh}_{\text{En}}(e_{\mathcal{G}}) \cdot t_{\leq})) \\
&= O(|G|^2 \cdot o_{\rightarrow} \cdot (|G|^{d-1})^2 \cdot (d^2 + d + |G|^{d-1} \cdot d)) \\
&= O(o_{\rightarrow} \cdot |G|^{2d+2-2} \cdot (d^2 + |G|^{d-1} \cdot d)) \\
&= O(o_{\rightarrow} \cdot |G|^{2d} \cdot (d^2 + |G|^{d-1} \cdot d))
\end{aligned}$$

For space complexity, we obtain:

$$\begin{aligned}
& O(|G| \cdot \text{wdh}_{\text{En}}(e_{\mathcal{G}}) \cdot d) \\
&= O(|G| \cdot |G|^{d-1} \cdot d) \\
&= O(|G|^d \cdot d) \quad \square
\end{aligned}$$

Because of the parameterization, we can easily see that some complexities go away if we use a more abstract energy lattice.

Definition 4.13 (Flattened energies). For d -dimensional energy games, we define the *flattened energies* as $\widehat{\text{En}} := (\{0, 1, \infty\})^d$. A standard energy $e \in \text{En}$ is cast to $\widehat{\text{En}}$ by \hat{e} where

$$(\hat{e})_k := \begin{cases} e_k & \text{if } e_k \leq 1 \\ \infty & \text{otherwise} \end{cases}.$$

We denote as *flattened energy game winner problem* the variant of Problem 2 that outputs

$$\widehat{\text{Win}}_{\text{a}}^{\min}(g) := \text{Min}(\text{Win}_{\text{a}}(g) \cap \widehat{\text{En}}).$$

Algorithm 4.1 can still be used to solve the flattened version of the problem by adapting $\text{upd}^{\mathcal{S}}$ to represent all components above 1 by ∞ . Effectively, this decouples the size of Pareto fronts from the game size. In this instance, Lemma 4.5 becomes:

Lemma 4.6 (Flattened complexity). *Algorithm 4.1 solves the flattened energy game problem on a d -dimensional declining energy game $\mathcal{G}^{\blacktriangleright} = (G, G_{\text{d}}, \rightarrow, w)$ in time*

$$O(o_{\rightarrow} \cdot |G|^{2 \cdot d} \cdot (d^2 + 3^{d-1} \cdot d))$$

and in space $O(|G| \cdot 3^{d-1} \cdot d)$.

The bounds we have established are *exponential* with regard to the dimensionality d . But in our use case, we care for energy games of fixed dimensionality. The time bounds are *polynomial* with regard to game graph size. The space bounds behave similarly, but, in the *flattened* variant, fixed energy makes the space usage even drop to be *linear* in terms of game graph size.

We have gathered enough material to prove that the spectroscopy problem for the P-easy strong spectrum is indeed solvable in polynomial time, thus justifying the name we chose for the spectrum.

4.3.5 Polynomial Spectroscopy Complexity

As outlined in Section 3.3.1, we consider the spectroscopy problem for p and q solved when we can compute $\text{Min}(\mathbf{N} \setminus \mathbf{N}_{p,q})$. More specifically, according to Theorem 4.1, the defender-won energies answer the spectroscopy question of Problem 1:

$$\mathbf{N}_{p,q}^{\text{peasy}} = \text{Win}_d^{\mathcal{G}_B^\bullet}([p, q]).$$

By the determinacy of winning budgets (Proposition 4.2),

$$\mathbf{N}^{\text{peasy}} \setminus \mathbf{N}_{p,q}^{\text{peasy}} = \text{Win}_a^{\mathcal{G}_B^\bullet}([p, q])$$

and thus

$$\text{Min}(\mathbf{N}^{\text{peasy}} \setminus \mathbf{N}_{p,q}^{\text{peasy}}) = \text{Win}_a^{\min \mathcal{G}_B^\bullet}([p, q]).$$

Thereby, Problem 1 for $\mathbf{N}^{\text{peasy}}$ on a transition system reduces to Problem 2 on the corresponding bisimulation energy game \mathcal{G}_B^\bullet . The latter is answered by Algorithm 4.1.

Theorem 4.4 (Overall complexity). *Given a transition system \mathcal{S} , the spectroscopy problem for the $\mathbf{N}^{\text{peasy}}$ -spectrum can be solved in polynomial time with respect to the size of \mathcal{S} .*

Proof. Theorem 4.1 has established that we can solve the spectroscopy problem for the $\mathbf{N}^{\text{peasy}}$ -spectrum by deciding the winning budgets of the bisimulation energy game \mathcal{G}_B^\bullet on $\mathcal{S} = (\mathcal{P}, \text{Act}, \rightarrow)$. Thus, all that remains to be done is to instantiate the winning budget complexity of Lemma 4.5 for the case $d = 3$ with the size of \mathcal{G}_B^\bullet according to Definition 4.8.

The number of attacker positions is bounded by $|\mathcal{P}|^2$. These positions can initiate up to $|\mathcal{P}| \cdot |\rightarrow|$ simulation challenges, leading to a similarly-bounded number of defender positions, which again can be left by $|\mathcal{P}| \cdot |\rightarrow|$ moves.

In total, this amounts to $o_{\rightarrow_B}^\bullet$ in $O(|\mathcal{P}| \cdot |\rightarrow|)$ and to $|G_B^\bullet|$ in $O(|\mathcal{P}|^2)$. Inserting the parameters in the time bounds of Lemma 4.5 yields:

$$\begin{aligned} & O(\quad o_{\rightarrow_B}^\bullet \quad \cdot \quad |G_B^\bullet|^{2 \cdot d} \quad \cdot \quad (d^2 + |G_B^\bullet|^{d-1} \cdot d) \quad) \\ &= O(\quad (|\mathcal{P}| \cdot |\rightarrow|) \quad \cdot \quad (|\mathcal{P}|^2)^{2 \cdot 3} \quad \cdot \quad (3^2 + (|\mathcal{P}|^2)^{3-1} \cdot 3) \quad) \\ &= O(\quad (|\mathcal{P}| \cdot |\rightarrow|) \quad \cdot \quad |\mathcal{P}|^{12} \quad \cdot \quad |\mathcal{P}|^4 \quad) \\ &= O(\quad |\mathcal{P}|^{17} \cdot |\rightarrow| \quad). \end{aligned}$$

For space complexity, the approach arrives at $O((|\mathcal{P}| \cdot |\rightarrow|)^3)$. The bound drops to $O(|\mathcal{P}| \cdot |\rightarrow|)$ in the flattened variant of Lemma 4.6. \square

4.4 Discussion

This chapter has shown how to solve the spectroscopy problem for the $\mathbf{N}^{\text{peasy}}$ spectrum in polynomial time, thereby deciding all its preorders and equivalences *at once*.

The energy game approach. The core ingredient has been to characterize the spectrum’s observation languages through an energy game (Idea 6, Theorem 4.1) derived from the bisimulation game. The Pareto front view is slightly more general than other characteristic games. For instance, Hüttel & Shukla (1996) use reachability games to establish polynomial upper bounds individually for simulation-like notions.

The focus on the attacker in the bisimulation game is highly analogous to Geuvers & Jacobs (2021) championing *apartness* as the dual of bisimilarity to work with equivalences. As Keiren & Willemse (2024) show, apartness proofs and attacker strategies correspond.

To compute the Pareto fronts, we use a fixed point algorithm (Idea 7, Algorithm 4.1). It can be thought of as a generalization of well-known ways to compute shortest distances in graphs (Mohri, 2002).

Remark 4.4 (Shortest paths). The energy game problem of this chapter can be seen as a *generalization of the shortest path problem* on directed graphs to account for two players as well as for much more general distances and cost functions.

To see how, consider energy games on (\mathbb{N}, \leq) where only deadlock positions belong to the defender and where all updates constitute one-dimensional \mathbb{N} -subtraction. Then, $e \in \text{Win}_a^{\min}(g)$ iff e is the length of a shortest path from g to some defender position (where negative $(-n)$ -updates stand for what is usually written as a positive edge weight n for graphs). In this instance, Algorithm 4.1 behaves like the well-known Bellman–Ford algorithm for shortest paths.

What’s new? Algorithm 4.1 contains relevant generalizations and practical improvements compared to the one originally used in Bisping (2023b). Most importantly, the declining energy version loses a dimension of exponentiality. For the polynomial equivalences, we arrive at a reasonably polynomial complexity for the spectroscopy instantiation, including reasonable space complexity for a flattened energy lattice. Thanks to Lemke’s work Lemke (2024), we have a thorough Isabelle/HOL formalization.

What’s next? At this point, we have, in principle, *answered our main research question* how to conveniently decide for a pair of states which notions from a spectrum of behavioral equivalences relate the two. In the second half of the thesis, we will leverage this approach to first treat the “strong spectrum” of concrete equivalences and then the “weak spectrum” of equivalences that abstract internal behavior.

Knowing from Section 3.3.3 that parts of the strong spectrum are PSPACE-hard, the algorithmic complexity indirectly proves that the bisimulation game cannot be used to characterize the whole spectrum. (Unless the polynomial hierarchy miraculously collapses to $\text{PSPACE} = \text{P}$.)

Thus, “*we’re gonna need a bigger boat*” game, at least exponentially-sized, to also cover PSPACE-hard notions.

5 Spectroscopy of the Strong Equivalence Spectrum

The energy game approach can, in fact, be leveraged to solve the spectroscopy problem for the *whole* of the strong equivalence spectrum.

The core ingredient of this chapter is a richer energy game, we dub “spectroscopy game,” to be presented in Section 5.1. The bisimulation energy game of Chapter 4 is not complex enough to capture all strong equivalences of Chapter 3. It is “too easy” for the attacker in the sense that they can use unbounded conjunctions to account for every transition the defender might choose after each observation. But the weaker notions of the spectrum heavily limit how many conjunctions may be used to name a distinction!

Our core trick will be to break the link between conjunctions and observation sequences in the game, analogously to the *subset construction* on finite automata.

Related publications. This chapter showcases the main result of my paper “Process equivalence problems as energy games” (Bisping, 2023b). We move beyond Bisping (2023b) by also deriving individual equivalence games from our characterization and showing how these enable polynomial-time decision procedures for P-easy notions.

i Idea 8: Subsets to separate observations and conjunctions

Trace-like notions can be addressed in the spectroscopy game by interjecting a *subset construction* on defender-controlled states between moves that correspond to observations and to conjunctions.

This will lead to exponential blow-up. Some of it can be alleviated, as we will see in Section 5.2, but most is necessary when deciding strong notions collectively. In Section 5.3, we discuss how to get rid of the blow-up where possible when instantiating the game to *decide equivalences individually*.

i Idea 9: Generating equivalence checkers

By instantiating the spectroscopy game with known initial energies and exploiting the supply of conjunction moves, we can derive efficient decision procedures for individual equivalences.

Figure 5.1 gives an overview of the two routes to decide equivalences collectively and individually that we will explore.

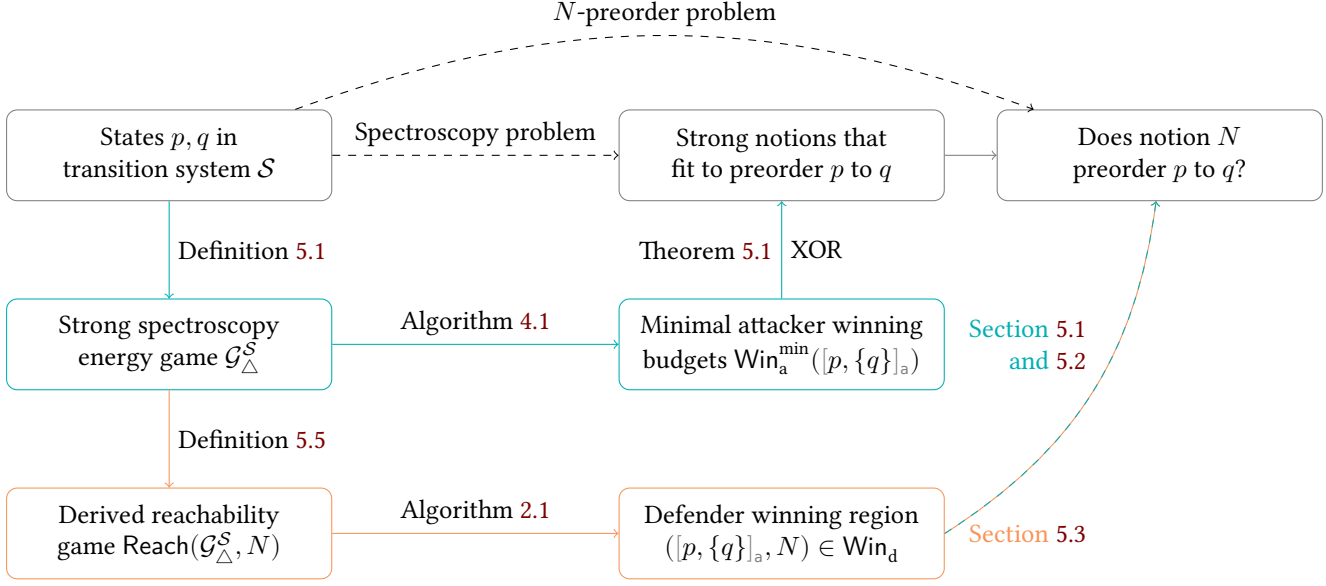


Figure 5.1: How we apply the spectroscopy approach to the full strong spectrum.

5.1 The Strong Spectroscopy Game

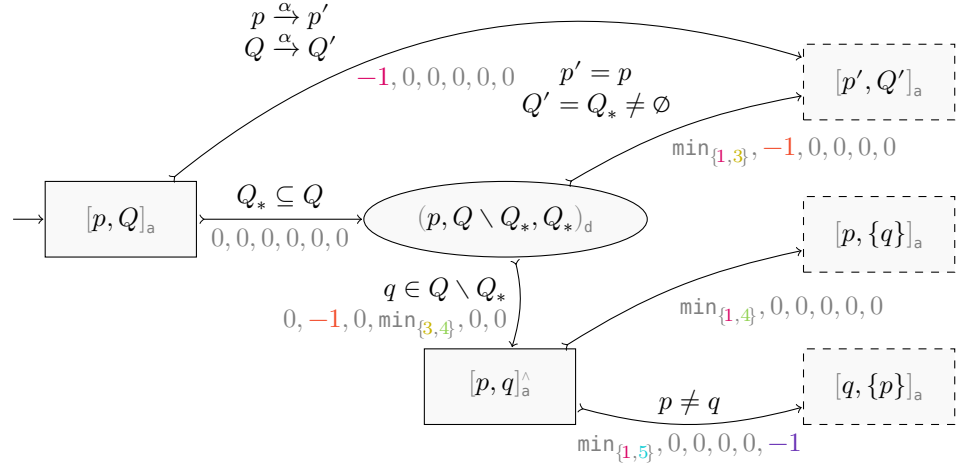
In this section, we examine a game that does not “overlook” trace observations, as the *bisimulation energy game* (Definition 4.8) does.

Example 5.1 (Without a trace). Consider the CCS processes $a.b.a$ and $a.(a + b)$. Clearly, only the first one allows the trace observation $\langle a \rangle \langle b \rangle \langle a \rangle$. Thus, the processes are not bisimilar and can be distinguished in the bisimulation game. But the bisimulation game formula (cf. Lemma 2.9) derivable from the game would be $\langle a \rangle \wedge \{ \langle b \rangle, \langle b \rangle \wedge \{ \langle a \rangle \} \}$, which is not part of trace observations $\mathcal{O}^{\text{strong}}_{(\infty, 0, 0, 0, 0, 0)}$.

It might be possible to reconstruct trace observations from such tree-like observations as the one of Example 5.1. But by the closing arguments of Section 4.4, and also according to Martens & Groote (2023), this must inherently be NP-hard. What we will do instead is adapt the bisimulation game such that it also expresses trace-like distinctions, and precisely counts conjunctions.

5.1.1 The Game

The *strong spectroscopy game* is played not on pairs of states, but on a pair of a state and a *set of states*! Figure 5.2 gives a graphical representation. The intuition is that the attacker shows how to construct formulas that distinguish a process p from every q in a set of processes Q . As long as observations happen, the defender is moved to the set Q of states that are reachable through

Figure 5.2: Schematic spectroscopy game \mathcal{G}_Δ of Definition 5.1.

nondeterminism; only at conjunction moves does the defender have to choose a $q \in Q$. Treating conjunctions more explicitly allows us to track the depth of positive conjuncts in sufficient detail.

Again, energies limit the syntactic expressiveness of the formulas.

- The first dimension bounds for how many turns the attacker may challenge observations of actions.
- The second dimension limits how often they may use conjunctions to resolve nondeterminism.
- The third, fourth, and fifth dimensions limit how deeply observations may nest underneath a conjunction, for which they do not need to change themselves. The third stands for one of the deepest positive conjuncts and the fourth for the other positive conjuncts; the fifth stands for negative conjuncts.
- The last dimension limits how often the attacker may use negations to enforce symmetry by swapping sides.

The moves of the following definition closely match productions in the grammar of observations for the strong spectrum of Definition 3.7.

Definition 5.1 (Strong spectroscopy game). For a system $\mathcal{S} = (\mathcal{P}, \text{Act}, \rightarrow)$, the 6-dimensional *strong spectroscopy game* $\mathcal{G}_\Delta^{\mathcal{S}} = (G, G_d, \rightarrow, w)$ consists of

- *attacker (main) positions* $[p, Q]_a \in G_a$,
- *attacker conjunct positions* $[p, q]_a^w \in G_a$,
- *defender conjunction positions* $(p, Q, Q_*)_d \in G_d$,

where $p, q \in \mathcal{P}$ and $Q, Q_* \in 2^{\mathcal{P}}$, and six kinds of moves:

<i>observation</i>	$[p, Q]_a$	$\xrightarrow{-1, 0, 0, 0, 0, 0}$	$[p', Q']_a$	if $p \xrightarrow{\alpha} p', Q \xrightarrow{\alpha} Q'$,
<i>conjunction</i>	$[p, Q]_a$	$\xrightarrow{0, 0, 0, 0, 0, 0}$	$(p, Q \setminus Q_*, Q_*)_d$	if $Q_* \subseteq Q$,
<i>conj. revival</i>	$(p, Q, Q_*)_d$	$\xrightarrow{\min\{1, 3\}, -1, 0, 0, 0, 0}$	$[p, Q_*]_a$	if $Q_* \neq \emptyset$,
<i>conj. answer</i>	$(p, Q, Q_*)_d$	$\xrightarrow{0, -1, 0, \min\{3, 4\}, 0, 0}$	$[p, q]_a^\wedge$	if $q \in Q$,
<i>positive conjunct</i>	$[p, q]_a^\wedge$	$\xrightarrow{\min\{1, 4\}, 0, 0, 0, 0, 0}$	$[p, \{q\}]_a$	and
<i>negative conjunct</i>	$[p, q]_a^\wedge$	$\xrightarrow{\min\{1, 5\}, 0, 0, 0, 0, -1}$	$[q, \{p\}]_a$	if $p \neq q$.

On the processes of Example 5.1, the attacker would move:

$$\begin{aligned}
 [a.b.a, \{a.(a+b)\}]_a &\xrightarrow{-\hat{e}_1} [b.a, \{a, b\}]_a \\
 &\xrightarrow{-\hat{e}_1} [a, \{0\}]_a \\
 &\xrightarrow{-\hat{e}_1} [0, \emptyset]_a \\
 &\xrightarrow{0} (0, \emptyset, \emptyset)_d \not\vdash
 \end{aligned}$$

Thereby, the attacker would get the defender stuck on a trace budget of $(3, 0, 0, 0, 0, 0) \leq (\infty, 0, 0, 0, 0, 0) = T$.

The handling of conjunctions can be a little more intricate.

Example 5.2 (Failure traces enter the game). Let us think back to Example 3.5, that is, to $Q' := \tau.(a.a + b.b)$, $T'_{aa} := Q' + \tau.a.a$, and $T'_a := Q' + \tau.a$, where T'_{aa} is distinguished from T'_a by the failure-trace observation $\langle \tau \rangle \wedge \{\langle a \rangle \langle a \rangle, \neg \langle b \rangle\} \in \mathcal{O}_{(3,1,2,0,1,1)}^{\text{strong}} \subseteq \mathcal{O}_{\text{FT}}^{\text{strong}}$.

To point out a similar distinction in the spectroscopy game, the attacker moves via τ :

$$\begin{aligned}
 [T'_{aa}, \{T'_a\}]_a &\xrightarrow{-\hat{e}_1} [a.a, \{a, a.a + b.b\}]_a \\
 &\xrightarrow{0} (a.a, \{a.a + b.b\}, \{a\})_d
 \end{aligned}$$

Now, the defender has two options.

- If defender chooses conjunction answer $\dots \xrightarrow{0, -1, 0, \min\{3, 4\}, 0, 0} [a.a, a.a + b.b]_a^\wedge$, then attacker points out that b is not possible for the left process:

$$\begin{aligned}
 [a.a, a.a + b.b]_a^\wedge &\xrightarrow{\min\{1, 5\}, 0, 0, 0, 0, -1} [a.a + b.b, \{a.a\}]_a \\
 &\xrightarrow{-\hat{e}_1} [b, \emptyset]_a \\
 &\xrightarrow{0} (b, \emptyset, \emptyset)_d \not\vdash
 \end{aligned}$$

- If defender opts for a conjunction revival $\dots \xrightarrow{\min\{1, 3\}, -1, 0, 0, 0, 0} [a.a, \{a\}]_a$, then attacker highlights that the left process can do aa :

$$\begin{aligned}
 [a.a, \{a\}]_a &\xrightarrow{-\hat{e}_1} [a, \{0\}]_a \\
 &\xrightarrow{-\hat{e}_1} [0, \emptyset]_a \\
 &\xrightarrow{0} (0, \emptyset, \emptyset)_d \not\vdash
 \end{aligned}$$

To win in both cases, the attacker must start with an energy budget of at least $(3, 1, 2, 0, 1, 1)$.

The game also allows an alternative through nested negations that the attacker wins at $(3, 2, 0, 0, 2, 2)$, but this is not interesting for specific equivalences below bisimilarity.

Example 5.3 (Half-simulated philosophers). Let us see what our prototype implementation on equiv.io reports about our original example processes P and Q of Chapter 2. We can input the philosopher processes of Example 2.2 and Example 2.3,⁸¹ obtaining a transition system matching Example 2.1:

```
PA = fork.a
PB = fork.b
P  = (fork! | PA | PB) \ {fork}

Q  = (fork! | fork.(a + b)) \ {fork}

@compare P, Q
```

Starting @compare (on equiv.io) triggers a spectroscopy along the lines we have discussed, leading to the output:⁸²

- Preordered by:
 - simulation
- Left-right-distinguished by:
 - $\langle \tau \rangle \wedge \{\neg \langle b \rangle \top\}$ (failure)
 - $\langle \tau \rangle \wedge \{\neg \langle a \rangle \top\}$ (failure)
- Equated by:
 - trace

That P is simulated by Q matches our finding from Example 2.6. Internally this is established by building the spectroscopy game and computing that $\text{Win}_a^{\min}([P, \{Q\}]_a) = \{(2, 1, 0, 0, 1, 1)\}$.

That no equivalence besides or above simulation can hold, is justified to the user by the failure $\langle \tau \rangle \wedge \{\neg \langle a \rangle \top\}$, which we also discussed in Section 3.1.2. In the upcoming Section 5.1.2, we will also give the rules that the tool uses to construct this witness in Strat_Δ .

Invoking @compare Q, P for the other direction, equiv.io reports $\langle \tau \rangle \wedge \{\langle b \rangle \top, \langle a \rangle \top\}$ as a distinguishing formula disproving simulation (as we have found in Example 2.12).

The spectroscopy game still is a bisimulation game in the following sense.

Proposition 5.1 (Defender bisimilarity). *States p_0 and q_0 are bisimilar precisely if the defender wins \mathcal{G}_Δ from $[p_0, \{q_0\}]_a$ for every initial energy budget e_0 , that is, if $(\infty, \infty, \infty, \infty, \infty, \infty) \in \text{Win}_d([p_0, \{q_0\}]_a)$.*⁸³

Remark 5.1 (The Δ -symbol). We use the Δ in \mathcal{G}_Δ with a double meaning: To symbolize a *prism*, which reveals the spectrum of light, and to mean *difference*, as the game expresses an abstract form of subtraction in the sense of Section 3.3.2.

⁸¹ A usage guide on the syntax can be found in Section 8.1.1.

[Interactive model on equiv.io](https://equiv.io).

⁸² A spectroscopy game on this example can also be played through in the computer game *The Spectroscopy Invaders*, see Section 8.2.1.

⁸³ Proof in Lemma 1 of Bisping (2023c).

5.1.2 Correctness

To prove correctness, we proceed as with the bisimulation energy game in Section 4.2.3: On the one hand, we show that game moves correspond to formulas of similar prices and that the attacker winning implies the formulas to be distinguishing. On the other hand, we establish that formulas of certain expressiveness prices certify winning attacks of matching budgets.

Definition 5.2 (Strategy formulas for \mathcal{G}_Δ). In the context of a strong spectroscopy game \mathcal{G}_Δ^S , the set of *strategy formulas* $\text{Strat}_\Delta(g, e)$ for a game position g and a budget e is defined inductively by the following rules in Figure 5.3.

$$\begin{array}{l}
\text{observation} \frac{[p, Q]_a \xrightarrow{-\hat{e}_1} [p', Q']_a \quad e' = \text{upd}(-\hat{e}_1, e) \in \text{Win}_a^{\mathcal{G}_\Delta}([p', Q']_a) \quad p \xrightarrow{\alpha} p' \quad Q \xrightarrow{\alpha} Q' \quad \varphi \in \text{Strat}_\Delta([p', Q']_a, e')}{\langle \alpha \rangle \varphi \in \text{Strat}_\Delta([p, Q]_a, e)} \\
\\
\text{conjunction} \frac{[p, Q]_a \xrightarrow{\alpha} (p, Q', Q_*)_d \quad e \in \text{Win}_a^{\mathcal{G}_\Delta}((p, Q', Q_*)_d) \quad \varphi \in \text{Strat}_\Delta((p, Q', Q_*)_d, e)}{\varphi \in \text{Strat}_\Delta([p, Q]_a, e)} \\
\\
\text{conj. answer} \frac{\forall q \in Q. (p, Q, \emptyset)_d \xrightarrow{u_q} [p, q]_a^\wedge \wedge e_q = \text{upd}(u_q, e) \in \text{Win}_a^{\mathcal{G}_\Delta}([p, q]_a^\wedge) \wedge \psi_q \in \text{Strat}_\Delta([p, q]_a^\wedge, e_q)}{\bigwedge_{q \in Q} \psi_q \in \text{Strat}_\Delta((p, Q, \emptyset)_d, e)} \\
\\
\text{conj. revival} \frac{\forall q \in Q. (p, Q, Q_*)_d \xrightarrow{u_q} [p, q]_a^\wedge \wedge e_q = \text{upd}(u_q, e) \in \text{Win}_a^{\mathcal{G}_\Delta}([p, q]_a^\wedge) \wedge \psi_q \in \text{Strat}_\Delta([p, q]_a^\wedge, e_q) \quad (p, Q, Q_*)_d \xrightarrow{u_*} [p, Q_*]_a \quad e_* = \text{upd}(u_*, e) \in \text{Win}_a^{\mathcal{G}_\Delta}([p, Q_*]_a) \quad \psi_* = \langle \alpha \rangle \varphi \in \text{Strat}_\Delta([p, Q_*]_a, e_*)}{\bigwedge_{q \in Q \cup \{*\}} \psi_q \in \text{Strat}_\Delta((p, Q, Q_*)_d, e)} \\
\\
\text{positive conjunct} \frac{[p, q]_a^\wedge \xrightarrow{u} [p, \{q\}]_a \quad e' = \text{upd}(u, e) \in \text{Win}_a^{\mathcal{G}_\Delta}([p, \{q\}]_a) \quad \langle \alpha \rangle \varphi \in \text{Strat}_\Delta([p, \{q\}]_a, e')}{\langle \alpha \rangle \varphi \in \text{Strat}_\Delta([p, q]_a^\wedge, e)} \\
\\
\text{negative conjunct} \frac{[p, q]_a^\wedge \xrightarrow{u} [q, \{p\}]_a \quad e' = \text{upd}(u, e) \in \text{Win}_a^{\mathcal{G}_\Delta}([q, \{p\}]_a) \quad p \neq q \quad \langle \alpha \rangle \varphi \in \text{Strat}_\Delta([q, \{p\}]_a, e')}{\neg \langle \alpha \rangle \varphi \in \text{Strat}_\Delta([p, q]_a^\wedge, e)}
\end{array}$$

Figure 5.3: Rules to derive distinguishing formulas from winning attack strategies.

The base case of the definition is the rule for *conjunction answers* at $(p, \emptyset, \emptyset)_d$. It yields the strategy formula \top , which trivially distinguishes any p from the empty set.

The correctness proof for \mathcal{G}_Δ proceeds basically as with the bisimulation energy game \mathcal{G}_B^\bullet , establishing soundness as in Lemma 4.1 and completeness as in Lemma 4.2. Proofs can be found in Bisping (2023c). There are two minor definitional nuances between the presentation here and there, upon which we will comment in Remark 5.2.

Lemma 5.1 (Distinction soundness). *If $e \in \text{Win}_a^{\mathcal{G}_\Delta}([p, Q]_a)$, then there is $\varphi \in \text{Strat}_\Delta([p, Q]_a, e)$ with $\varphi \in \mathcal{O}_e^{\text{strong}}$, $p \in \llbracket \varphi \rrbracket$ and $Q \cap \llbracket \varphi \rrbracket = \emptyset$.*⁸⁴

⁸⁴ Proof in Lemma 2, 3, and 4 of Bisping (2023c).

Lemma 5.2 (Distinction completeness). *If there is $\varphi \in \mathcal{O}_N^{\text{strong}}$ with $p \in \llbracket \varphi \rrbracket$ and $Q \cap \llbracket \varphi \rrbracket = \emptyset$, then $N \in \text{Win}_a^{\mathcal{G}_\Delta}([p, Q]_a)$.*⁸⁵

⁸⁵ Proof in Lemma 5 of Bisping (2023c).

Taken together, Lemma 5.1 and Lemma 5.2 prove that the strong spectroscopy game \mathcal{G}_Δ precisely characterizes the strong equivalence spectrum $\mathbf{N}^{\text{strong}}$.

Theorem 5.1 ($\mathbf{N}^{\text{strong}}$ -characterization). *$p \preceq_{\mathcal{O}_N} q$ for $N \in \mathbf{N}^{\text{strong}}$ precisely if the defender wins \mathcal{G}_Δ for the attacker starting from $[p, \{q\}]_a$ with budget N .*

Thus, we can solve the spectroscopy problem (Problem 1) by computing $\mathbf{N}_{p,q}^{\text{strong}} = \mathbf{N}^{\text{strong}} \setminus \text{Win}_a^{\mathcal{G}_\Delta}([p, \{q\}]_a)$.

Remark 5.2 (Differences to conference version). As mentioned in Remark 3.1, Bisping (2023b) uses a slightly different pricing, where $\text{expr}(\top) = \hat{e}_2$. This difference is reflected in the game by Definition 5.1 charging $-\hat{e}_2$ for conjunctions *after* the defender conjunction positions instead of *before*. With this, the attacker wins $[p, \emptyset]_a \xrightarrow{\circ} (p, \emptyset, \emptyset)_d$ with budget 0, while the original definition in Bisping (2023b) would cost them \hat{e}_2 for $[p, \emptyset]_a \xrightarrow{-\hat{e}_2} (p, \emptyset, \emptyset)_d$.

Also, Bisping (2023b) defines the spectrum coordinates by giving the expr function instead of an $\mathcal{O}_{N \in \mathbf{N}}$ hierarchy.

The conference paper results apply to the present presentation *mutatis mutandis*. However, we will achieve better complexity results in the next section, thanks to Lemke (2024).

5.1.3 Complexity

As expected, the bigger spectroscopy game leads to exponential runtimes. This comes as no surprise, for we have already discussed in Section 3.3.3 that the spectroscopy problem on the whole strong spectrum is PSPACE-hard.

Theorem 5.2 (Strong spectroscopy complexity). *Given a transition system \mathcal{S} , the spectroscopy problem for the $\mathbf{N}^{\text{strong}}$ -spectrum can be solved by the game approach in exponential time and space with respect to the state space size $|\mathcal{P}|$.*

Proof. According to Theorem 5.1, we can solve the spectroscopy problem for the $\mathbf{N}^{\text{strong}}$ -spectrum by deciding the winning budgets of the strong spectroscopy game \mathcal{G}_Δ^S on $\mathcal{S} = (\mathcal{P}, \text{Act}, \rightarrow)$. We instantiate the winning budget complexity of Lemma 4.5 for the case $d = 6$ with the size of \mathcal{G}_Δ according to Definition 5.1.

The number of attacker main positions is bounded by $|\mathcal{P}| \cdot 2^{|\mathcal{P}|}$. The number of conjunction moves and defender conjunction positions is bounded by $|\mathcal{P}| \cdot 3^{|\mathcal{P}|}$.

The maximal out-degree for attacker positions is in $O(2^{|\mathcal{P}|})$, which dominates that of defender conjunction positions of $O(|\mathcal{P}|)$,

This leads to o_{\rightarrow} in $O(2^{|\mathcal{P}|})$ and to $|G_{\Delta}|$ in $O(|\mathcal{P}| \cdot 3^{|\mathcal{P}|})$. Filling in the time bounds of Lemma 4.5 yields:

$$\begin{aligned}
& O(\quad o_{\rightarrow} \quad \cdot \quad |G|^{2 \cdot d} \quad \cdot \quad (d^2 + |G|^{d-1} \cdot d) \quad) \\
= & O(\quad 2^{|\mathcal{P}|} \quad \cdot \quad (|\mathcal{P}| \cdot 3^{|\mathcal{P}|})^{2 \cdot 6} \quad \cdot \quad (6^2 + (|\mathcal{P}| \cdot 3^{|\mathcal{P}|})^{6-1} \cdot 6) \quad) \\
= & O(\quad 2^{|\mathcal{P}|} \quad \cdot \quad |\mathcal{P}|^{12} \cdot 3^{12|\mathcal{P}|} \quad \cdot \quad |\mathcal{P}|^5 \cdot 3^{5|\mathcal{P}|} \quad) \\
= & O(\quad |\mathcal{P}|^{17} \quad \cdot \quad 2^{|\mathcal{P}|} \cdot 3^{17|\mathcal{P}|} \quad).
\end{aligned}$$

For space complexity, the approach arrives at $O(|\mathcal{P}|^6 \cdot 3^{6|\mathcal{P}|})$. \square

Still, there are ways to decrease the complexity of the algorithm and thus increase applicability. We will explore these in the next section.

5.2 Clever Games on Subgraphs

We can exploit properties of the equivalences to focus on simpler variants of the spectroscopy game.

1. Properties of bisimilarity allow to reduce the game graph size without losing information.
2. Since the specific coordinates in Figure 3.8 only use 0, 1, and ∞ in components, we can employ Lemma 4.6 to slightly improve complexity bounds.
3. With more specific equivalences in mind, we can also focus on subgraphs of the spectroscopy game \mathcal{G}_{Δ} in order to obtain better bounds.

Section 5.2.1 will first discuss how parts of the game graph can be pruned thanks to properties of bisimilarity. Then, we will use all listed tricks in the “clever” strong spectroscopy game in Section 5.2.2. In the subsequent Section 5.3, we will take the last point to the extreme by instantiating the spectroscopy game down to only decide single equivalences.

5.2.1 Pruning with Logic

We can exploit that the spectroscopy game is a bisimulation game by Proposition 5.1 to reduce game graph size. In particular, we will profit from the properties of bisimilarity being symmetric and transitive.

Lemma 5.3 (Symmetry defense). *On a strong spectroscopy game, $p \in Q$ implies $\text{Win}_a([p, Q]_a) = \emptyset$.*

Proof. This is a contrapositive consequence of Lemma 5.1: As soon as Q contains a state bisimilar to p , there cannot be a distinguishing formula for them by Theorem 2.1. This is the case here because $p \in Q$ and $p \sim_B p$ by Lemma 2.1. Thus, there cannot be $e \in \text{Win}_a([p, Q]_a)$. \square

For the game graph, this means that we do not need to consider any outgoing moves of $[p, \{p\} \cup Q]_a$ -positions. They will not lead to attacker wins anyway. All game graph parts that are only reached through such positions can be disregarded.

Another important trick to reduce game size is to first apply bisimulation-minimization to the transition system and then solely construct the spectroscopy game for the minimized system, $\mathcal{G}_{\Delta}^{S/\sim_B}$. This approach is sound by the following lemma.

Lemma 5.4 (Quotienting by bisimilarity). *The attacker wins $[p, \{q\}]_a$ on \mathcal{G}_{Δ}^S with e precisely if they win $[[p]_{\sim_B}]_a, \{[q]_{\sim_B}\}$ on $\mathcal{G}_{\Delta}^{S/\sim_B}$.*

Proof. By Proposition 2.4, $p \sim_B [p]_{\sim_B}$ and $q \sim_B [q]_{\sim_B}$ on the merger of S and S/\sim_B . Therefore, the pairs each fulfill the same HML formulas by Theorem 2.1, which implies equal distinguishing formula sets. We can use Lemma 5.1 and Lemma 5.2 to translate these sets to and from the winning budgets. \square

Example 5.4 (Some measurements). Both tricks of pruning “symmetric” parts (Lemma 5.3) and pre-minimization by bisimilarity (Lemma 5.4) make an effective difference for the applicability of the spectroscopy approach:


 **tool.benchmark.Sizemark** uses a small example system from Bisping (2023b) modelling Peterson’s mutual exclusion algorithm.⁸⁶ The system has only $|\mathcal{P}| = 20$ states, but a lot of nondeterminism due to saturation with internal behavior.

Figure 5.4 lists the sizes of game graphs on this system. The default spectroscopy game at the top has roughly 1200 thousand game positions.

Pruning symmetric parts removes roughly four fifths of the graph (leaving 241 thousand positions).

Only applying the bisimulation minimization roughly halves the graph (leaving 661 thousand positions). The bisimulation reduction is surprisingly effective, given that the bisimulation minimized Peterson system still has 19 states, that is, only one fewer than the original system. On the other hand, bisimilar states are precisely those that allow the biggest space of distinctions, blowing up the game.

Interestingly, the combination of both tricks reduces the game graph to 67 thousand positions, roughly 5 % of the original size. (If the effects of both minimizations were only multiplicative, the number would be around 133 thousand states.) This over-proportionate effect on the one hand has to do with the general exponentialities of the game graph, but also with the fact that bisimulation minimization increases the instances where pruning is applicable.⁸⁷

The implementations (Chapter 8) sometimes use small additional tricks for pruning. For instance, one can leave out observation moves behind $[p, \emptyset]_a$ positions and finish the game optimally by conjunction.

⁸⁶ The reported numbers are produced by sbt "shared/run sizemark --strong-game".

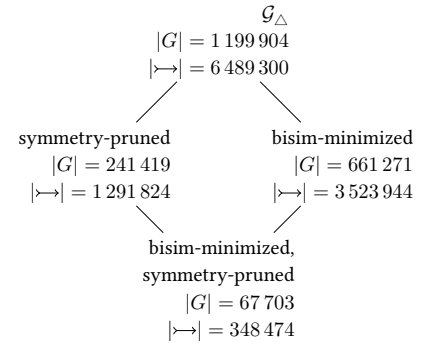


Figure 5.4: Example sizes of a spectroscopy game with and without pruning on Peterson’s mutex system.

⁸⁷ This interplay can already be seen on the bisimulation game. Figure 2.15 contains a big region after $[t_{ab}, q_{ab}]$ and $[t_{ab}, q_{ab}]$. Bisimulation minimization of the input system would shrink this region as $t_{ab} \sim_B q_{ab}$. The two positions would be merged into $[\{t_{ab}, q_{ab}\}, \{t_{ab}, q_{ab}\}]$. But this is a symmetric position! Therefore, we can drop this part altogether.

So far, the optimizations have been *lossless* in the sense that we do not lose any information on the distinguishability of processes. If we are willing to lose precision, additional reductions of the game graph are possible.

5.2.2 The Clever Game

We can *become more clever by looking one step ahead*.

The spectroscopy game \mathcal{G}_Δ of Definition 5.1 may branch exponentially with respect to $|Q|$ at conjunction challenges after $[p, Q]_a$. For an efficient implementation, it is desirable to not do that.

Given the spectrum we are interested in, we can drastically limit the sensible attacker moves to four options by a little lookahead into the enabled actions $\text{Ini}(q)$ of $q \in Q$ and $\text{Ini}(p)$.

In the following, we will focus on the lower-resolution sub-spectrum $\widehat{\mathbf{N}}^{\text{strong}} \subseteq \{0, 1, \infty\}^6 \subseteq \mathbf{N}^{\text{strong}}$, where dimensions 3, 4, and 5 of modal depth for revivals, positive and negative conjuncts can only appear in certain combinations.

Definition 5.3 (Simpler strong spectrum). The *simpler strong spectrum*, $\widehat{\mathbf{N}}^{\text{strong}}$, is defined as

$$\widehat{\mathbf{N}}^{\text{strong}} := \{N \in \{0, 1, \infty\}^6 \mid N_4 \leq N_3 \wedge (N_5 = \infty \longrightarrow N_3 = N_4)\}.$$

Observe that all the coordinates of the strong linear-time–branching-time spectrum (Figure 3.8) are still covered by the simpler spectrum.

Definition 5.4 (Clever spectroscopy game). The *clever spectroscopy game* $\mathcal{G}_\blacktriangle$, is defined exactly like the previous spectroscopy game of Definition 5.1, with the restriction of the conjunction challenges

$$[p, Q]_a \xrightarrow{0} \blacktriangle (p, Q \setminus Q_*, Q_*)_d \quad \text{with } Q_* \subseteq Q,$$

to situations where Q_* is one of the four choices

- \emptyset – no revivals,
- $\{q \in Q \mid \text{Ini}(q) \subseteq \text{Ini}(p)\}$ – enabledness-dominated revivals,
- $\{q \in Q \mid \text{Ini}(p) \subseteq \text{Ini}(q)\}$ – enabledness-dominating revivals, or
- $\{q \in Q \mid \text{Ini}(p) = \text{Ini}(q)\}$ – enabledness-matching revivals.

The idea here is that the attacker only thinks about certain partitionings: Those where the Q_* -revival deals with parts of Q that cannot be discharged through failure or readiness observations. The usage of $\text{Ini}(\cdot)$ is comparable to a small look-ahead into the possibilities to win through observations inside the conjunction.

Theorem 5.3 (Correctness of cleverness). For $N \in \widehat{\mathbf{N}}^{\text{strong}}$, the attacker wins $\mathcal{G}_\blacktriangle$ from $[p_0, Q_0]_a$ with energy N precisely if they win \mathcal{G}_Δ from $[p_0, Q_0]_a$ with energy N .⁸⁸

⁸⁸ Proof in Theorem 2 of Bisping (2023c).

Theorem 5.4 (Complexity of cleverness). *Given a transition system $\mathcal{S} = (\mathcal{P}, \text{Act}, \rightarrow)$, the spectroscopy problem for the simpler $\widehat{\mathcal{N}}^{\text{strong}}$ -spectrum is solved by the game approach in $O(|\mathcal{P}|^{13} \cdot 2^{12|\mathcal{P}|})$ time and $O(|\mathcal{P}| \cdot 2^{|\mathcal{P}|})$ space,*

Proof. We instantiate the “flattened” winning budget complexity of Lemma 4.6 for the case $d = 6$ with the size of $\mathcal{G}_\blacktriangle$ according to Definition 5.4.

Now, the number of all game positions is bounded by $|G_\blacktriangle| \in O(|\mathcal{P}| \cdot 2^{|\mathcal{P}|})$.

Concerning the maximal out-degree, now, defender conjunction positions are dominant, $o_{\rightarrow\blacktriangle} \in O(|\mathcal{P}|)$. Inserting the parameters in the time bounds of Lemma 4.6 yields:

$$\begin{aligned} & O(\quad o_{\rightarrow\blacktriangle} \quad \cdot \quad |G_\blacktriangle|^{2-d} \quad \cdot \quad (d^2 + 3^{d-1} \cdot d)) \\ = & O(\quad (|\mathcal{P}|) \quad \cdot \quad (|\mathcal{P}| \cdot 2^{|\mathcal{P}|})^{12} \quad \cdot \quad (6^2 + 3^5 \cdot 6) \quad) \\ = & O(|\mathcal{P}|^{13} \cdot 2^{12|\mathcal{P}|}). \end{aligned}$$

For space complexity, only game positions are relevant, $O(|G_\blacktriangle| \cdot 3^{d-1} \cdot d) = O(|\mathcal{P}| \cdot 2^{|\mathcal{P}|})$. \square

Example 5.5 (More measurements). Let us see how the “clever game” adds up with the pruning effects on the Peterson example of Example 5.4.

Figure 5.5 is the lower part of a cube of possible game graph optimizations, where Figure 5.4 has already provided the upper part.

We can directly observe that $\mathcal{G}_\blacktriangle$ is smaller than \mathcal{G}_\triangle by a factor of more than 1000, with 1020 positions instead of 1 199 904. This is due to the high nondeterminism of the Peterson system, which leads to big Q -sets in $[p, Q]_a$ -positions. These entail exponentially many conjunction moves in \mathcal{G}_\triangle , which are absent in the $\mathcal{G}_\blacktriangle$ subgraph.

The tricks of bisimulation minimization and pruning still pay off, reducing the game positions to 65 % of the unpruned game. This is way less effective than in Example 5.4 (where this factor has been 5 %), because the savings are not boosted by exponentialities.

Still, it makes sense to combine all three optimizations. All in all, we have reduced the number of positions by a factor of 1800, and the number of moves by factor 2700.

Especially, the cleverness has paid off. Without these tricks, the approach would not be applicable to systems with relevant nondeterminism due to communication.⁸⁹

5.3 Deciding Individual Equivalences

As we have seen, the strong spectroscopy characterizes the spectroscopy problem *and* each individual equivalence (and preorder). This section is about the feasibility to decide individual preorders using the spectroscopy game.

The core idea behind Section 5.3.1 is to *derive a reachability game* according to Definition 4.4, starting with an energy level from the strong spectrum,

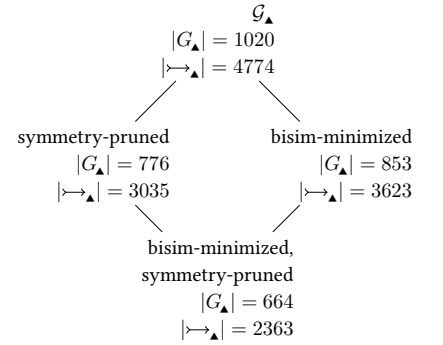


Figure 5.5: Sizes of *clever* spectroscopy games with and without pruning on Peterson’s mutex system.

⁸⁹ The blow-up on Peterson’s mutex is precisely the argument I made in the introduction of Bisping (2023b) for why one has to move beyond the original spectroscopy algorithm of Bisping et al. (2022). Therefore, the original/deprecated approach also is not present in this thesis.

Definition 3.8. Then, Algorithm 2.1 can be employed to decide player winning regions of the game and thus the equivalence.

Of course, in this situation, one wants to avoid complexities that are part of the spectroscopy game only to capture other equivalences. Section 5.3.2 will *regain polynomiality* for the P-easy portion of equivalence problems.

5.3.1 Deriving Equivalence Games

Let us first explicate how to characterize equivalences through reachability games derived according to Definition 4.4.

Definition 5.5 (Derived strong equivalence games). Given a system $\mathcal{S} = (\mathcal{P}, \text{Act}, \rightarrow)$, a strong notion of equivalence $N \in \mathbf{N}^{\text{strong}}$, and a pair of states p, q , the *derived N -preorder game* $\mathcal{G}_{\Delta N}^{\mathcal{S}, p, q}$ is defined as the part of the reachability game $(\mathcal{G}_{\Delta}^{\mathcal{S}})^R$ derived from the spectroscopy game under starting position $([p, \{q\}]_a, N)$.

We thus receive a game characterization for every equivalence, generalizing Stirling's result of the bisimulation game from Theorem 2.2.

Theorem 5.5 (Parametric characterizations). $p \preceq_{\mathcal{O}_N} q$ precisely if the defender wins the N -preorder game $\mathcal{G}_{\Delta N}^{\mathcal{S}, p, q}$ from $([p, \{q\}]_a, N)$.

Proof. By Theorem 5.1, $p \preceq_{\mathcal{O}_N} q$ implies that the defender wins $\mathcal{G}_{\Delta}^{\mathcal{S}}$ from $[p, \{q\}]_a$ when the attacker starts at energy N . In other words, $N \notin \text{Win}_a([p, \{q\}]_a)$ for $\mathcal{G}_{\Delta}^{\mathcal{S}}$. Lifted via Proposition 4.4, this implies $([p, \{q\}]_a, N) \notin \text{Win}_a$ for the derived reachability game $(\mathcal{G}_{\Delta}^{\mathcal{S}})^R$. By Definition 5.5, the defender thus wins $\mathcal{G}_{\Delta N}^{\mathcal{S}, p, q}$. As the proof steps work in both directions, this completes the proof. \square

The same reasoning works with the clever game version $\mathcal{G}_{\blacktriangle}$, restricted to the $\hat{\mathbf{N}}^{\text{strong}}$ -spectrum used in Theorem 5.3.

Using Algorithm 2.1, our results entail decision procedures for each individual equivalence of the strong spectrum. Of course, it makes sense for algorithms to employ the clever game and pruning.

Example 5.6 (Checking equivalences). The following processes are two of the more complex ones of the separating examples by van Glabbeek (2001, p. 59). They are equal with respect to ready traces, but not for simulation-like and possible-future-like equivalences.

[Interactive model on equiv.io.](#)

```

ABCACB = a.(b.d + c.e) + a.(c.f + b.g)
ABC     = a.(b.d + c.e + c.f + b.g)

@check trace,          ABCACB, ABC
@check ready-trace,    ABCACB, ABC
@check simulation,     ABCACB, ABC

```

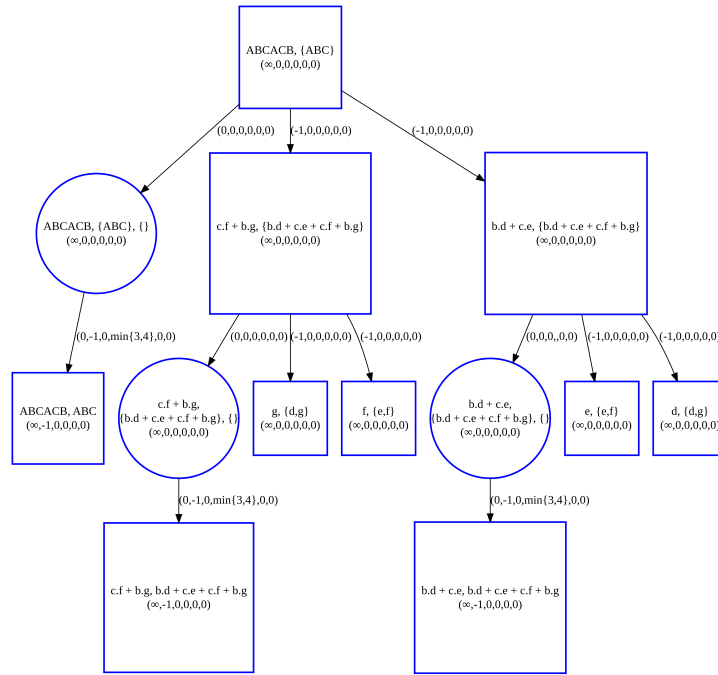


Figure 5.6: Pruned clever strong spectroscopy game below position $[ABCACB, \{ABC\}]_a$ instantiated to starting energy $T = (\infty, 0, 0, 0, 0, 0)$ to decide trace equivalence.

```
@check impossible-future, ABCACB, ABC
@check bisimulation,      ABCACB, ABC
```

The `@check` keyword performs individual comparisons on equiv.io. Invoking the check for traces, for instance, yields the following output:

```
@check trace, ABCACB, ABC
- "States are equivalent."
```

The T-preorder game derived from the (pruned clever) spectroscopy game under $[ABCACB, \{ABC\}]_a$ is depicted in Figure 5.6. Each position is won by the defender (hinted at by the positions being colored in blue).

Interestingly, the derived reachability game of Figure 5.6 has *fewer* positions than the pruned clever spectroscopy game below $[ABCACB, \{ABC\}]_a$ would have.⁹⁰

As the derived game positions are obtained through products of original positions and energies, one might assume the derived preorder games to be way bigger than the pure spectroscopy game. But for the named notions, this is usually not the case because of two aspects:

The first important fine point is that many notions have ∞ -entries and that $\infty - 1 = \infty$. For instance, in the derived bisimulation game, starting

⁹⁰ Incidentally, the answer to the question how many positions this spectroscopy game would have is 42.

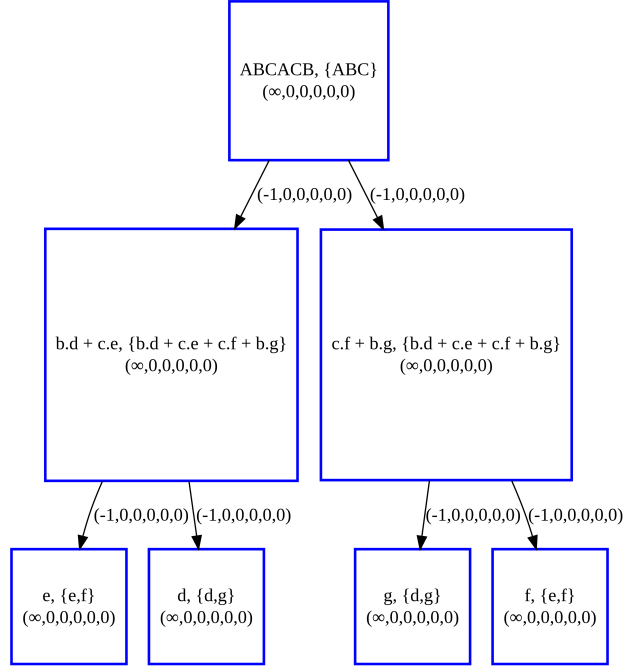


Figure 5.7: The subgraph of Figure 5.6 that actually is used in [equiv.io](#), disregarding some attacker options.

at $(\infty, \infty, \infty, \infty, \infty, \infty)$, all positions will remain at this energy level. Thus, the derived bisimulation game $\mathcal{G}_{\Delta B}^{S,p,q}$ has isomorphic positions to \mathcal{G}_{Δ} under $[p, \{q\}]_a$.

The other point to consider is that the game is cut off at exhausted energies. Therefore, 0-entries in components lead to smaller games, only using subgraphs of the spectroscopy game.

Only 1-components grow the derived game graph. In principle, each component that starts at 1 instead of ∞ or 0 might roughly double the graph size.

The [equiv.io](#) implementation does prune some attacker moves that cannot be winning, in particular non-empty conjunctions if the conjunction budget is at 0. Therefore, the subgraph used by the tool for Example 5.6 in reality is the one in Figure 5.7.

Still, nondeterminism in the transition system might lead to exponentially-sized derived games. Especially in the case of bisimilarity and similarity, such a blow-up is not necessary. As Chapter 4 has shown, they should be P-easy.

5.3.2 Regaining Polynomiality in Derived Games

So far, the derived games include the exponentiality due to the subset construction. But often, we can restrict the attacker to only use a polynomially-

sized portion of the game. If they have energy for unbounded conjunctions, this restriction will not affect their wins.

Proposition 5.2 (Defender restriction). *If the attacker wins $([p, Q]_a, e)$ in a derived preorder game $\mathcal{G}_{\Delta N}$ or $\mathcal{G}_{\mathbf{A}N}$, they also win $([p, Q']_a, e)$ for all $Q' \subseteq Q$.*

Proof. If the attacker wins $[p, Q]_a$ with e in a spectroscopy game, they also win $[p, Q']_a$ with e for all $Q' \subseteq Q$. This is a side benefit of the spectroscopy characterization (Lemma 5.1 and 5.2) through distinctions, as $Q \cap \llbracket \varphi \rrbracket = \emptyset$ implies $Q' \cap \llbracket \varphi \rrbracket = \emptyset$ for any φ . We can lift this result immediately to the derived preorder games as in Theorem 5.5. \square

Lemma 5.5 (Conjunctive attacks). *If the attacker wins $([p, Q]_a, e)$ with $|Q| > 1$ in a derived preorder game with $e \geq (0, \infty, \infty, \infty, 0, 0)$, then they can also win without using any observation moves of the form $([p, Q]_a, e) \rightsquigarrow ([p', Q']_a, e')$.*

Proof. Any observation move $([p, Q]_a, e) \rightsquigarrow ([p', Q']_a, e')$ must be due to a transition $p \xrightarrow{\alpha} p'$ with $Q \xrightarrow{\alpha} Q'$. Instead of the observation, the attacker can move to a conjunction $([p, Q]_a, e) \rightsquigarrow ((p, Q, \emptyset)_d, e)$. After the defender choice of $q \in Q$, the game continues with the same energy $\dots \rightsquigarrow ([p, q]_a, e) \rightsquigarrow ([p, \{q\}]_a, e) \rightsquigarrow ([p', \text{Der}(q, \alpha)]_a, e')$, thanks to the sufficient energy budget. As $\text{Der}(q, \alpha) \subseteq Q'$, we can use Proposition 5.2. \square

Thus the closure of a logic under conjunctions allows us to use a game that, practically, leaves out many formulas, without losing distinctiveness. This trick is basically the same as in the crucial $\langle \alpha \rangle$ -case of Lemma 2.7 that “game HML” $\mathcal{O}_{[B]}$ and HML are equally distinctive.

As a consequence, we can obtain a *polynomial complexity bound* for notions that do not restrict conjunctions and positive conjuncts.

Lemma 5.6 (Polynomial equivalence checking). *For a strong notion $N \in \{0, 1, \infty\}^6$ with $N \geq (0, \infty, \infty, \infty, 0, 0)$, and a system $(\mathcal{P}, \text{Act}, \rightarrow)$, one can decide $p \preceq_{\mathcal{O}_N} q$ in polynomial time, by computing who wins the N -preorder game in $O(|\mathcal{P}| \cdot |\rightarrow|)$ time and space.*

Proof. We can use the clever version of the strong spectroscopy game $\mathcal{G}_{\mathbf{A}}$ to decide individual equivalences for these coordinates by Theorem 5.5 and Theorem 5.3. For $N \geq (0, \infty, \infty, \infty, 0, 0)$, this also works on the subgraph where nondeterminism is resolved immediately thanks to Lemma 5.5.

We want to use that, according to Proposition 2.9, deciding a reachability game $(G, G_d, \rightsquigarrow)$ takes $O(|\rightsquigarrow|)$ time and $O(|G|)$ space.

We observe that the $\{0, 1, \infty\}$ -valued energy components in the game might only create a constant-factor increase in the size of $\mathcal{G}_{\mathbf{A}N}$, compared to the underlying game $\mathcal{G}_{\mathbf{A}}$.⁹¹ Therefore, the following arguments focus on the latter.

There are at most $|\mathcal{P}|^2$ positions of the form $[p, \{q\}]_a$. Whenever they reach $[p', Q']_a$ with $|Q'| > 1$, Lemma 5.5 allows us to prune away all observation moves deriving from there, and return to an attacker main position with

⁹¹ If we were to allow k -step equivalences as in Figure 4.5, k would also become a factor.

$|Q''| \leq 1$. The intermediate $[p', Q']_a$ -positions are a function of initial q and the transitions from p , and thus collectively bounded by $O(|\mathcal{P}| \cdot |\rightarrow|)$, which bounds attacker positions as a whole.

In the clever spectroscopy game \mathcal{G}_Δ , there is a maximum of only five defender positions per such attacker position. Also, the subsequent attacker conjunct positions, bounded by $|\mathcal{P}|^2$, do not affect the bound.

For the moves, only observation moves and conjunct answers can impact complexity. The observations are bounded in $O(|\mathcal{P}| \cdot |\rightarrow|)$, and the conjunct answers in $O(|\mathcal{P}|)$, which is dominated by the first.

Inserting our bounds in Proposition 2.9, we obtain time and space complexity of $O(|\mathcal{P}| \cdot |\rightarrow|)$.⁹² \square

Example 5.7 (Measurements of derived games). Let us examine how the instance game sizes behave on the games of Example 5.6! Figure 5.8 maps out the sizes of game instantiations for the preorders of the strong spectrum as in Figure 3.8.⁹³ Each game is derived from the clever spectroscopy game, with pruning and with capped nondeterminism where possible; to enable equivalence checks, the games combine the parts under $[ABCACB, \{ABC\}]_a$ and under $[ABC, \{ABCACB\}]_a$. The game size is defined as the sum of positions and moves, $|\mathcal{G}| = |G| + |\rightarrow|$. Red parts symbolize bigger game graphs.

The data match our thoughts on game sizes in Section 5.3.1. For instance, around the trace game $\mathcal{G}_{\Delta T}$, we can observe that zero-components usually reduce the game graph size. Between mutual simulation $\mathcal{G}_{\Delta IS}$ and bisimulation $\mathcal{G}_{\Delta B}$, one can see that ∞ -components and 0-components are usually better than 1-components. The game for ready traces $\mathcal{G}_{\Delta RT}$, where all dimensions play a role, is the most costly.

5.4 Discussion

With this chapter, we have broadened the spectroscopy game approach of Chapter 4 to account for the full strong spectrum of Chapter 3, thereby solving its spectroscopy problem (Problem 1).

One game, many notions. The critical ingredient has been to *cover trace-like observations* through a *subset construction* on states (Idea 8). This construction can also be viewed game-theoretically as a way to model an attacker with *imperfect information* about the exact choices the defender makes in simulating steps. The conjunction and negation points then describe moves where the attacker can obtain this information. Fahrenberg & Legay (2014) employ this alternative view in their game description for quantitative variants of strong equivalences.

All prior publications with generalized game characterizations of the spectrum describe a hierarchy of reachability games—as opposed to our view of *one energy game per system*.

The *general idea of giving generalized games* for the strong spectrum first appears in Shukla et al. (1996). It receives a fuller treatment by Chen & Deng

⁹² This is with the understanding that one can ignore storage space for the positions: One can use pointers to Q' -sets in the attacker and defender positions, and not store them verbatim. This saves the bound as there can only be $|\rightarrow|$ such sets of size $|Q'| > 1$. So, overall, these sets also will take up $O(|\mathcal{P}| \cdot |\rightarrow|)$ space.

⁹³ The reported numbers are produced via `sbt "shared/run eqchecks --strong-game"` for the pair L50/R50 by [tool.benchmark.LTBTSEquivalenceChecks](https://github.com/ltbtse/tool.benchmark.LTBTSEquivalenceChecks).

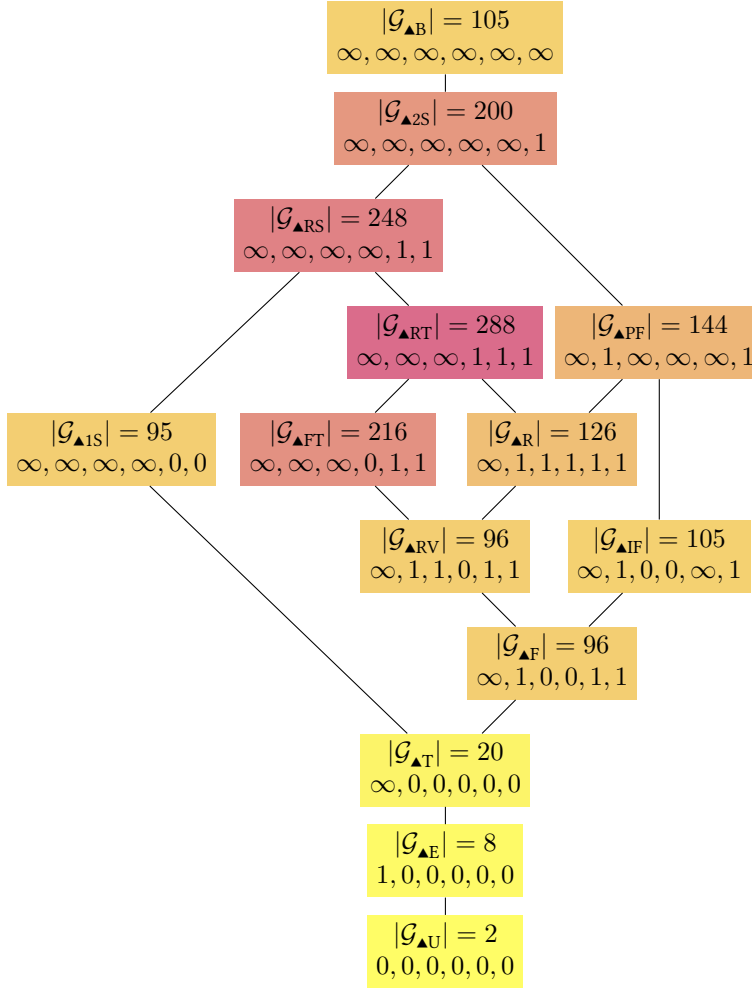


Figure 5.8: Instance game sizes of Example 5.7.

(2008). Both publications, however, use word-transition moves for trace-like equivalences. In algorithms, this is unfortunate, since most interesting systems allow infinitely long transition sequences: For finite-state systems, this infinity appears as soon as there is a transition cycle. Therefore, the subset construction (or imperfect-information view) is technologically superior. On the side of theory, the link between moves in the game and conjunctions in modal logics adds clarity to our construction. It reinforces the idea of Section 3.3.2 that equivalence checking is, in a way, subtraction of satisfied formulas.

Difference to original formulation. This chapter's presentation improves in several ways on my original spectroscopy games. The idea to alternate *state subsets for nondeterministic words and conjunction moves to eliminate the blow-up* is introduced by Bisping & Nestmann (2021). Our initial 2021 game is

unable to correctly handle failure-traces and similar observations, which has been corrected in Bisping et al. (2022), thanks to Jansen’s input. With respect to Bisping et al. (2022), the presentation in this chapter has three advantages:

1. Bisping et al. (2022) uses a reachability game, on which we construct distinguishing formulas. The formulas are then pruned on-the-fly to select those with minimal syntactic prices. Introduced by Bisping (2023b), we now can perform the whole *spectroscopy on energy vectors*, thereby making costly explicit formula construction optional.
2. Bisping et al. (2022)’s game has super-exponential out-degree for conjunction moves. With the present formulation (from Bisping, 2023b), we can achieve constant degree, using the cleverness of Section 5.2.2.
3. On a pure reachability game, one cannot easily instantiate to select the subgraphs for individual equivalences as in Section 5.3.

Back to many games. The point of *deriving equivalence games from the energy game* (Idea 9) connects the one-energy-game approach back to the hierarchy-of-games approach of prior publications. With Section 5.3, the present chapter has moved beyond the scope of Bisping (2023b). Interestingly, Section 5.3.2 shows that, by prioritizing conjunctions in derived simulation-like games, one obtains $O(|\mathcal{P}| \cdot |\rightarrow|)$ complexity, restoring the P-easiness of the preceding chapter. For strong similarity, this matches the best known time bound, as discussed in Section 3.3.3. For bisimilarity, however, partition refinement outperforms the game approach.

We could go a step further in the polynomially-sized derivatives of the spectroscopy game, and “bake together” the attacker main positions after observations with the following defender positions. This would (almost) arrive at the game structure of last chapter’s bisimulation energy game! Conceptionally, the following relation emerges: The bisimulation game \mathcal{G}_B of Chapter 2 is a *shadow* of the bisimulation energy game \mathcal{G}_B^\star of Chapter 4, which in turn is a *shadow* of the strong spectroscopy game \mathcal{G}_Δ of the present chapter.

The dimensions of energy games allow us to encode different games *within* one game. This is analogous to stereoscopic pictures, decodable by 3D glasses (Figure 5.9).

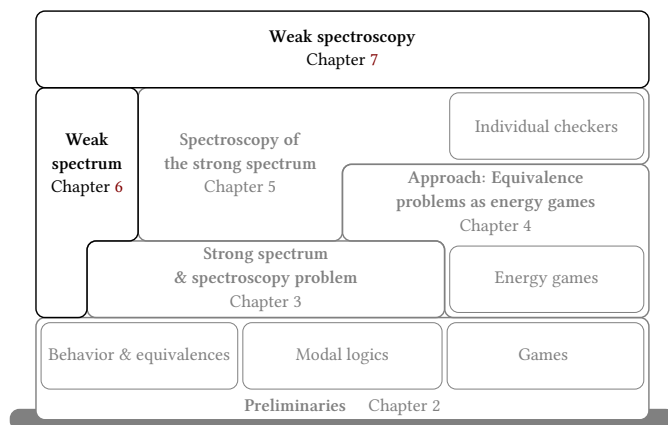


What's next? Continuing the trajectory of shadows, Part III of the thesis will abstract further what we have been doing. The strong spectrum of equivalences is usually too strong for the analysis of concurrent systems. Therefore, we will extend our approach to the *weak spectrum* of equivalences, accounting for the nuances of silent τ -steps. This is of double relevance: Practical applications of concurrency theory tend to call for weak semantics, and nobody has previously proposed generalized game characterizations for the weak spectrum, at all.

Figure 5.9: The author, discussing 3D energy games with participants of D-CON’25. (Photo: Nadine Karsten)

Part III

... of Concurrent Programs



6 Recharting the Weak Silent-Step Spectrum

Virtually all *applications* of concurrency theory use *weak* behavioral equivalences, which can equate systems in spite of differing internal activity. Therefore, it matters that we can lift our approach to also account for such equivalences.

This chapter serves as a *second round of preliminaries* and context needed for the spectrum of *weak equivalences*. Some things are more complicated in the weak spectrum. For instance, we will be discussing *eight* weak forms of bisimilarity.

At its heart, this chapter aims to find a modal spectrum characterization, where HML grammar and pricing are designed in such a way that we can apply the spectroscopy approach in the following Chapter 7. The *top element of the weak spectrum* will be the logic of *stability-respecting branching bisimilarity*.

This is where the thesis arrives at “The linear time–branching time spectrum II: The semantics of sequential systems with silent moves” (van Glabbeek, 1993), already featured in Figure 1.1 of the introduction. We will reframe a big chunk of it to fit our modal framework in Figure 6.5. In particular, we try to be much more restricted in the use of special modalities than the presentation of van Glabbeek (1993).

i Idea 10: Taking a sublogic to weaken HML

We can use a subset of Hennessy–Milner logic to characterize most interesting weak equivalences.

Section 6.1 provides some preliminaries and defines the HML subset HML_{SRBB} , which will correspond to the notion of stability-respecting branching bisimilarity. Section 6.2 illustrates the use of some of the weak notions through small case studies from compiler verification and concurrency theory. In Section 6.3, we unfold the hierarchy of modal characterizations for the weak spectrum.

Related publications. This chapter is based on parts of the draft “Deciding all behavioral equivalences at once II: The silent-step spectrum” (Bisping & Jansen, 2025) and of “Characterizing contrasimilarity through games, modal logic, and complexity” (Bisping & Montanari, 2024). For a fuller presentation of the spectrum of weak equivalences itself, van Glabbeek (1993) is the authoritative work. A gentler introduction to key concepts is provided by Sangiorgi (2012).

6.1 Weak Equivalences in General

For practical problems, the equivalences we have discussed so far usually are *too strong*. They notice where in a process the *internal action* τ happens, that is:

$$a \approx \tau.a \approx \tau.\tau.a \approx \tau.a.\tau \approx a.\tau$$

For real-world models, we want equivalences to disregard such kinds of internal behavior as “silent” when comparing processes, such that:

$$a \sim \tau.a \sim \tau.\tau.a \sim \tau.a.\tau \sim a.\tau$$

Equivalences with this feature are called *weak*, alluding to the fact that they are less distinctive than the “strong” equivalences that treat τ like any other action.

Figure 6.1 shows the (strong-bisimulation-minimal) transition system of the example processes we would like to equate in weak equivalences.

The basic principle is that weak equivalences should maintain that some internal behavior happening before or after a visible action does not make a difference from the point of view of an observer. But this idea leads into a lot of fine points.

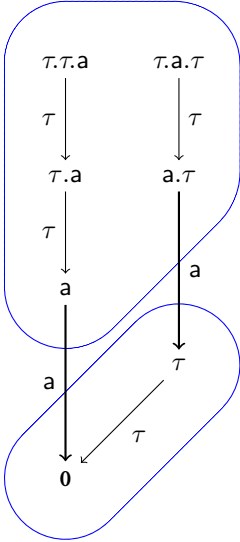


Figure 6.1: Processes that we would like to equate in weak equivalences (blue groups).

⁹⁴ 🧠 inductive [LTS.LTS_Tau.silent_reachable](#)

⁹⁵ 🧠 abbreviation [LTS.LTS_Tau.soft_step](#)

⁹⁶ 🧠 abbreviation [LTS.LTS_Tau.stable_state](#)

6.1.1 Silent Transitions

To help us with our abstractions of weak activity, we use some additional notation:

Definition 6.1 (Transition systems with internal activity). For a transition system $\mathcal{S} = (\mathcal{P}, Act, \rightarrow)$ with an *internal action* $\tau \in Act$, we call τ -transitions “silent” and define the following special transition relations plus HML modalities, where $Act_{\bullet} := Act \setminus \{\tau\}$ denotes “visible” actions:

Internal transition relation $p \twoheadrightarrow p'$ iff $p \xrightarrow{\tau^*} p'$, where τ^* is the reflexive transitive closure of silent steps.⁹⁴

The *internal* modality $\langle \varepsilon \rangle \varphi$ has $p \in \llbracket \langle \varepsilon \rangle \varphi \rrbracket$ iff there is $p' \in \llbracket \varphi \rrbracket$ with $p \twoheadrightarrow p'$.

Soft transition relation $p \xrightarrow{(\alpha)} p'$ iff $p \xrightarrow{\alpha} p'$, or if $\alpha = \tau$ and $p = p'$.⁹⁵

We accompany it by the *soft* observation modality $(\alpha)\varphi$ with $p \in \llbracket (\alpha)\varphi \rrbracket$ iff there is $p' \in \llbracket \varphi \rrbracket$ with $p \xrightarrow{(\alpha)} p'$.

Weak word steps $p \xrightarrow{\vec{w}} p'$ for $\vec{w} = (w_1 w_2 \dots w_n) \in Act_{\bullet}^*$ iff there is a path with $p_i \xrightarrow{w_{i+1}} p_{i+1}$ for each $0 \leq i < n$ such that $p_0 = p$ and $p_n \twoheadrightarrow p'$.

Stable state A state p is called *stable* iff $p \not\xrightarrow{\tau}$.⁹⁶

In HML, we can use $\neg \langle \tau \rangle \top$ to express stabilization.

Divergent state A state p is called *divergent* iff it allows an infinite sequence of τ -transitions, $p \xrightarrow{\tau}^\omega$.

Again, we implicitly lift the relations to sets of states, that is, for instance, $P \twoheadrightarrow P'$ (with $P, P' \subseteq \mathcal{P}$) iff $P' = \{p' \in \mathcal{P} \mid \exists p \in P. p \twoheadrightarrow p'\}$.

In upcoming definitions and facts, we use the convention that α continues to stand for elements of Act , while a comes from $Act_{\bullet} = Act \setminus \{\tau\}$.

Example 6.1 (Weak steps). In Figure 6.1, the state $\tau.\tau.a$ allows the following weak steps:

- Soft τ -transitions to itself and its immediate successor: $\tau.\tau.a \xrightarrow{(\tau)} \tau.\tau.a$ and $\tau.\tau.a \xrightarrow{(\tau)} \tau.a$. (But $\tau.\tau.a \not\xrightarrow{\tau} \tau.\tau.a$!)
- The internal transition $\tau.\tau.a \rightarrow a$. (On top of the internal transitions due to $\xrightarrow{(\tau)} \subseteq \rightarrow$.)
- The weak word transition $\tau.\tau.a \xRightarrow{a} 0$.
- Starting in a , all kinds of a -steps coincide: $a \xRightarrow{a} 0$ and $a \xrightarrow{a} 0$ just as $a \xrightarrow{a} 0$.

$a, a.\tau$ and 0 are stable. No state is divergent.

Remark 6.1 (Just notation). Note that the notation for HML operators $\langle \varepsilon \rangle \dots$ and $(\alpha) \dots$ introduced in Definition 6.1 does not affect the expressiveness of HML. To see why, assume notation for disjunction $\bigvee_{i \in I} \psi_i := \neg \bigwedge_{i \in I} \neg \psi_i$, and for n -time application of an operator. Then, $\llbracket \langle \varepsilon \rangle \varphi \rrbracket = \llbracket \bigvee_{n \in \mathbb{N}} \langle \tau \rangle^n \varphi \rrbracket$. Also, $\llbracket (\tau) \varphi \rrbracket = \llbracket \bigvee \{ \varphi, \langle \tau \rangle \varphi \} \rrbracket$ and $\llbracket (a) \varphi \rrbracket = \llbracket \langle a \rangle \varphi \rrbracket$.

On the other hand, note that our infinitary HML cannot express proper divergence $p \xrightarrow{\tau}^\omega$ on infinitary systems. This would demand an “observation of infinite depth.” This clashes with the well-foundedness, which is implied by the recursive grammar of Definition 2.11. On systems with finite branching degree, however, the possibility of unbounded τ -trees and infinite τ -chains coincides, and divergence can be captured modally by $\bigwedge_{n \in \mathbb{N}} \langle \tau \rangle^n$.

6.1.2 Weak Traces and Weak Bisimulation

It is quite straightforward how to lift traces from the strong setting of Definition 2.4 and Definition 2.5 to the weak setting: Allow internal \rightarrow -behavior in between.

Definition 6.2 (Weak traces, preorder, and equivalence). The set of *weak traces* of a state $\text{WeakTraces}(p) \subseteq Act_{\bullet}^*$ is defined as $\vec{w} \in \text{WeakTraces}(p)$ iff there is p' such that $p \xRightarrow{\vec{w}} p'$.^{97,98}


Two states are *weakly trace-preordered*, $p \preceq_{\text{WT}} q$, if $\text{WeakTraces}(p) \subseteq \text{WeakTraces}(q)$.⁹⁹ As before, if both directions are maintained, the states are called *weakly trace-equivalent*, $p \sim_{\text{WT}} q$.

The blue groups of states in Figure 6.1 are weakly trace equivalent. Moreover, they also are related by the stronger notions of weak bisimulation and simulation.

Definition 6.3 (Weak simulation, preorder and equivalences). A relation, $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$, is called a *weak simulation* if, for each $(p, q) \in \mathcal{R}$,

⁹⁷  abbreviation [LTS.LTS_Tau.weak_traces](#)

⁹⁸ The Isabelle theory has a slightly different definition that also allows τ s in the trace words, which do not necessarily stand for proper $\xrightarrow{\tau}$ -steps.

⁹⁹  definition [LTS.LTS_Tau.weakly_trace_preordered](#)

- if $p \xrightarrow{a} p'$ with $a \in Act_{\bullet}$, there is a q' with $q \xrightarrow{a} q'$ and $(p', q') \in \mathcal{R}$; and
- if $p \xrightarrow{\tau} p'$, there is a q' with $q \xrightarrow{\tau} q'$ and $(p', q') \in \mathcal{R}$.

Weak simulation preorder, weak simulation equivalence and weak bisimilarity follow analogously to Definition 2.7:

- p is *weakly simulated* by q , $p \preceq_{\text{WS}} q$, iff there is a weak simulation \mathcal{R} with $(p, q) \in \mathcal{R}$.
- p is *weakly similar* to q , $p \sim_{\text{WS}} q$, iff $p \preceq_{\text{WS}} q$ and $q \preceq_{\text{WS}} p$.
- p is *weakly bisimilar* to q , $p \sim_{\text{WB}} q$, iff there is a *symmetric* weak simulation \mathcal{R} (i.e. $\mathcal{R} = \mathcal{R}^{-1}$) with $(p, q) \in \mathcal{R}$.

All three weak equivalences maintain that the small example processes of the introduction to Section 6.1 are equal.

Example 6.2 (Weakly simulated philosophers). For the processes of Example 2.1 (repeated in Figure 6.2), the weak traces would be $\text{WeakTraces}(P) = \{(), a, b\} = \text{WeakTraces}(Q)$. Consequently, P and Q are weakly trace-equivalent, $P \sim_{\text{WT}} Q$.

In Example 2.6, we have observed that the two processes are not (strongly) similar because $Q \not\preceq_S P$ due to $\langle \tau \rangle \wedge \{\langle a \rangle, \langle b \rangle\}$. Due to the weakening, however, there is a weak simulation for this direction, namely: $\{(Q, P), (q_{ab}, P), (q_1, p_1), (q_2, p_2)\}$. Therefore, $P \sim_{\text{WS}} Q$! A mutual weak simulation \mathcal{R}_{PQ} to justify this is drawn in dashed blue in Figure 6.2.

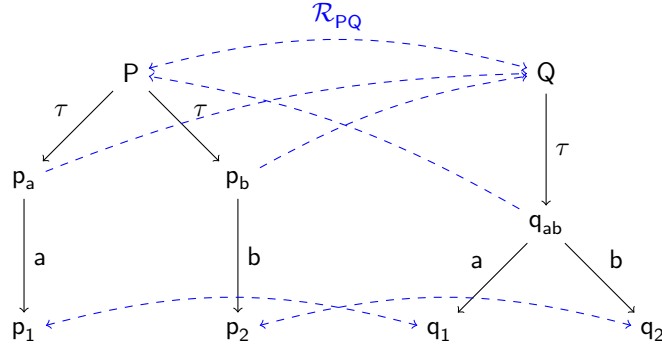


Figure 6.2: A mutual weak simulation on the original philosopher example.

However, P and Q are not weakly bisimilar. The reason is that the left process can weakly $\xrightarrow{\tau}$ -transition to a position where b is impossible, even if we would allow for some more internal $\xrightarrow{\tau}$ -behavior in between. In HML, this difference could be expressed by $\langle \varepsilon \rangle \neg \langle \varepsilon \rangle \langle b \rangle$ —we will turn to modal characterizations soon.

Definition 6.4 (Abstractions). We say that a weak notion of equivalence WN abstracts a strong notion N iff $\preceq_{WN} = \preceq_N$ for all transition systems without τ -transitions.

By this nomenclature, weak trace preorder, weak simulation preorder, and weak bisimilarity are abstractions of their strong counterparts from Chapter 2. But, as we will see in the next sections, there might be *multiple* abstractions of the same strong notion.

6.1.3 HML of Stability-Respecting Branching Bisimilarity

As noted in Remark 6.1, our HML notation for weak observations does not affect expressiveness—thus it still characterizes strong bisimilarity by Theorem 2.1. To characterize weak equivalences, we have to select appropriate subsets. We will use the following sublogic that, naturally, must be designed to correspond to the strongest weak notion we will be interested in.

Definition 6.5 (Branching Hennessy–Milner logic). We define *stability-respecting branching* Hennessy–Milner modal logic, HML_{SRBB} , over an alphabet of actions Act by the following context-free grammar starting with φ .¹⁰⁰

¹⁰⁰ 📦 datatype [HML_SRBB.hml_srbb](#)

$\varphi ::=$	$\langle \varepsilon \rangle \chi$		“delayed observation”
	$\bigwedge \{ \psi, \psi, \dots \}$		“immediate conjunction”
$\chi ::=$	$\langle a \rangle \varphi$	with $a \in \text{Act}$	“observation”
	$\bigwedge \{ \psi, \psi, \dots \}$		“standard conjunction”
	$\bigwedge \{ \neg \langle \tau \rangle \top, \psi, \psi, \dots \}$		“stable conjunction”
	$\bigwedge \{ (\alpha) \varphi, \psi, \psi, \dots \}$	with $\alpha \in \text{Act}$	“branching conjunction”
$\psi ::=$	$\neg \langle \varepsilon \rangle \chi \mid \langle \varepsilon \rangle \chi$		“negative / positive conjuncts”

We consider the semantics of HML_{SRBB} to be given by Definition 2.12 and Definition 6.1.

The name already alludes to HML_{SRBB} as a whole characterizing *stability-respecting branching bisimilarity*, which is a comparably strong abstraction of bisimilarity, a “strong weak bisimilarity,” so to speak.

Definition 6.6 (Stability). We call a relation \mathcal{R} *stability-respecting* iff, for each $(p, q) \in \mathcal{R}$ with $p \not\vdash$, there is some q' with $q \rightarrow q' \vdash$ and $(p, q') \in \mathcal{R}$.¹⁰¹

¹⁰¹ 📦 definition [LTS.LTS_Tau.stability_respecting](#)

Definition 6.7 (Branching bisimilarity). A symmetric relation \mathcal{R} is a *branching bisimulation* if, for all $(p, q) \in \mathcal{R}$, a step $p \xrightarrow{\alpha} p'$ implies (1) $\alpha = \tau$ and $(p', q) \in \mathcal{R}$, or (2) $q \rightarrow q' \xrightarrow{\alpha} q''$ for some q' and q'' with $(p, q') \in \mathcal{R}$ and $(p', q'') \in \mathcal{R}$.¹⁰²

¹⁰² 📦 definition [Branching_Bisimilarity.LTS_Tau.branching_simulation](#)

If there is a stability-respecting branching bisimulation $\mathcal{R}_{BB^{sr}}$ with $(p_0, q_0) \in \mathcal{R}_{BB^{sr}}$, then p_0 and q_0 are *stability-respecting branching bisimilar*.

The power of Definition 6.5 to distinguish mirrors exactly the power of Definition 6.7 to equate:

Theorem 6.1 (HML_{SRBB} Hennessy–Milner theorem). HML_{SRBB} characterizes *stability-respecting branching bisimilarity*, that is, there is a stability-respecting

¹⁰³ 📦 lemma [Branching_Bisimilarity.LTS_Tau_sr_branching_bisim_is_hmlsrbb](#)

branching bisimulation \mathcal{R} with $(p, q) \in \mathcal{R}$ precisely if there is no formula $\varphi \in \text{HML}_{\text{SRBB}}$ with $p \in \llbracket \varphi \rrbracket$ and $q \notin \llbracket \varphi \rrbracket$.¹⁰³

The paper proof in Bisping & Jansen (2025) proceeds quite similarly to other Hennessy–Milner theorems, as showcased in Theorem 2.1.

The example process groups of Figure 6.1 are stability-respecting branching bisimilar, as desired. But branching bisimilarity is more distinctive than weak bisimilarity:

Example 6.3 (The strength of branching). Compare $P_{ab} := a + \tau.b + b$ and a subgraph-variant $P_{a\tau b} := a + \tau.b$, where the b is *only* possible after committing to its branch. The processes are not branching bisimilar as $\langle \varepsilon \rangle \wedge \{ \langle b \rangle, \langle \varepsilon \rangle \langle a \rangle \}$ distinguishes them. Intuitively, the formula expresses that, right at the moment when b can happen, a is still (weakly) possible.

But the processes are weakly bisimilar, as weak bisimilarity does not care about branching due to τ -steps that can also be achieved by visible actions. A formal argument would be that $\text{id}_{\mathcal{P}} \cup \{ (P_{ab}, P_{a\tau b}), (P_{a\tau b}, P_{ab}) \}$ is a symmetric weak simulation.

Remark 6.2 (An equivalence indeed). It is a popular anecdote among those who know branching bisimilarity that, originally, a trivial proof of transitivity was “assumed,” but turned out not to exist. Six years after branching bisimilarity’s inception, “Branching bisimilarity is an equivalence indeed!” by Basten (1996) closed the gap with a proof that is surprisingly complex.

In our setting, the transitivity is an *immediate corollary* of having a modal characterization (Theorem 6.1).¹⁰⁴ This is one of the perks of modal characterizations, indeed (cf. Section 2.3.3).

6.2 Case Studies—and the Need for Other Weak Equivalences


In the world of weak equivalences, many things are a little more complicated than in the strong spectrum. We have already observed that bisimulation equivalence can be abstracted to branching bisimulation *and* weak bisimulation.

In this section, we use two minimal case study examples to see how one can arrive at even more abstractions of bisimulation and of failure equivalence.

Next chapter’s Section 7.2 will pick up on the examples to test that the spectroscopy algorithm can indeed simplify a researcher’s life.

6.2.1 Parallelizing Compilers—and Contrsimulation

Most of computer speed up in the last two decades has been due to parallelization of computation. Compilers will usually drift and parallelize sequential commands of a program. The claim behind such optimizations is that

¹⁰⁴  lemma `Branching_Bisimilarity.LTS_Tau`
`.sr_branching_bisimulated_transitive`

the communication behavior of the program stays the same.¹⁰⁵ So, naturally, the original program and the optimized program should be equivalent with respect to some notion of equivalence.

Bell (2014) ventures to prove that certain parallelizations of loops during compilation are sound. This runs into the problem that weak bisimilarity is too strong for this use case. The following example (adapted from Bell, 2013) shows why.

Example 6.4 (Parallelized execution). Consider the following sequential program P_{Seq} , which computes x (with possible values A and B), prints a header (output:) and then the computed value:

```
x = compute_A_or_B()
print("output:")
print(x)
```

A parallelization would find that the rendering of the header “output:” is independent of the computation of x and can thus be parallelized to P_{Para} :

```
x = compute_A_or_B() || print("output:")
print(x)
```

The `||` connector is supposed to mean a parallel execution of commands that synchronizes the branches at the end. In real-world programming languages, this would happen through spawning and joining subprocesses, future objects, or `async` segments.

Figure 6.3 shows the transition systems of the two programs. The τ -steps mark internal computation or synchronization. In particular, `compute_A_or_B` turns into a τ -step. The action `printO` stands for `print("output:")` announcing the coming output, and `printA` and `printB` are the values of x that are actually produced by `print(x)`.

Clearly, P_{Seq} and P_{Para} have the same weak traces. But they do not weakly simulate each other: P_{Para} can perform a $\xrightarrow{\text{printO}}$ -transition, immediately after which `printA` and `printB` are weakly possible at p_{ab} . The sequential P_{Seq} has no such states.

Bell (2013) proposes “eventual bisimilarity” as a one-off bisimulation-like notion to equate P_{Seq} and P_{Para} . In Bisping & Montanari (2024), we discuss how two more standard notions from the original spectrum neatly equate the processes: Stable bisimilarity and contrasimilarity.

Definition 6.8 (Stable bisimilarity). A *stable bisimulation* is a relation \mathcal{R} where, for all $(p, q) \in \mathcal{R}$ with $\vec{w} \in \text{Act}_{\bullet}$ and $p \xrightarrow{\vec{w}} p'$, there is a q' with $q \xrightarrow{\vec{w}} q'$, and, in case p' is stable, moreover $q' \not\vdash$ and $(p', q') \in \mathcal{R} \cap \mathcal{R}^{-1}$. The *stable bisimilarity* \sim_{SB} is defined as in Definition 6.3.

On the programs of Example 6.4, there is a stable bisimulation, connecting P_{Seq} and P_{Para} as well as the stable states with matching positions. Intuitively,

¹⁰⁵ In other regards, the program obviously differs. It is the *intention* to change aspects such as performance!

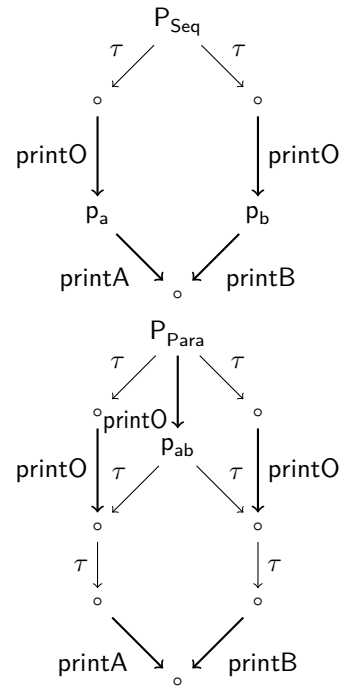


Figure 6.3: Transition systems representing P_{Seq} and P_{Para} .

stable bisimilarity is allowed to “skip” intermediate states that would break the bisimulation—such as p_{ab} in our example.

As we argue in Bisping & Montanari (2024), stable bisimilarity, in many ways, is better understood not as a stable variant of bisimilarity, but of *contrasimilarity*:

Definition 6.9 (Contrasimilarity). A *contrasimulation* is a relation \mathcal{R} where, for all $(p, q) \in \mathcal{R}$ with $\vec{w} \in \text{Act}_\bullet$ and $p \xrightarrow{\vec{w}} p'$, there is a q' with $q \xrightarrow{\vec{w}} q'$ and $(q', p') \in \mathcal{R}$.¹⁰⁶ The *contrasimulation preorder* \preceq_C and *contrasimilarity* \sim_C are defined as in Definition 6.3.

Contrasimilarity allows *local asymmetry* in matching internal transitions. A witness relation for Example 6.4 would also include (p_a, p_{ab}) and (p_b, p_{ab}) . The intuition is that contrasimulation preorder allows states to slightly get ahead of their counterparts— p_a is more committed than p_{ab} —as long as the latter can catch up silently.

Contrasimilarity and stable bisimilarity, both are at the weak end of abstractions of bisimilarity, “weakest” bisimilarities, one could say. We discuss them and their characterizations in great detail in Bisping & Montanari (2024).

Contrasimilarity and stable bisimilarity are nice for proofs as they have coinductive characterizations. They also are more well-behaved than Bell’s “eventual bisimilarity,” which fails to be an equivalence if the transition system has divergent states (cf. Bell, 2014, Section 3.3).

As we will establish in Section 7.2.1, contrasimilarity and stable bisimilarity indeed are the finest equivalences from the standard spectrum to equate the sequential and the parallel program.

Remark 6.3 (Half a definition of stable bisimilarity). Sangiorgi (2012, Section 6.5) gives the following, simpler definition for stable bisimulation:¹⁰⁷

A process relation \mathcal{R} is a *stable bisimulation* if, whenever $(p, q) \in \mathcal{R}$, for all \vec{w} we have:

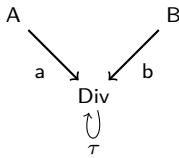
- for all p' with $p \xrightarrow{\vec{w}} p'$ and p' is stable, there is q' such that $q \xrightarrow{\vec{w}} q'$ and $(p', q') \in \mathcal{R}$,

and the converse on the actions from q .

Sangiorgi notes that the induced stable bisimilarity relation would not be transitive on transition systems with divergences. But the situation is even more severe!

Defined like this, stable bisimilarity would not even imply weak trace equivalence. Consider the processes A and B in the margin. As all their non-empty word transitions end in instable states, the variant from Sangiorgi (2012) ignores their differences altogether.

“Our” stable bisimilarity of Definition 6.8, however, does not suffer from this defect, for it demands matching of all word transitions. Our relational



¹⁰⁶ Note that p' and q' swap sides in the consequent!

¹⁰⁷ Quote adapted to the notation of the present thesis.

Definition 6.8 is informed by the upcoming modal characterizations of Section 6.3 (and of van Glabbeek (1993) and Bisping & Montanari (2024)). As discussed in Section 2.3.3, it is trivial to ensure that modally characterized relations are transitive and refine trace preorder. Such are the perks of modal characterizations!

6.2.2 Abstraction as Congruence—and Stable Failures

For trace-like equivalences, there commonly appear two kinds of abstractions in the literature: Ones where negation and conjunctive observations are only possible when processes have *stabilized*, that is, exhausted their possibilities of internal behavior, and ones where stability plays no role. We will illustrate these options using modal characterizations of stable and weak failure equivalence, abstracting Definition 3.2:

Definition 6.10 (Weak failure preorder and equivalence). Let *weak failure preorder and equivalence* be defined by the weak failure observations \mathcal{O}_{WF} produced by φ^{WF} in the following grammar, and *stable failure preorder and equivalence*, by the products of φ^{SF} .

$$\begin{aligned}\varphi^{\text{WF}} &::= \langle \varepsilon \rangle \langle a \rangle \varphi^{\text{WF}} \mid \langle \varepsilon \rangle \bigwedge_{i \in I} \neg \langle \varepsilon \rangle \langle a_i \rangle \top \\ \varphi^{\text{SF}} &::= \langle \varepsilon \rangle \langle a \rangle \varphi^{\text{SF}} \mid \top \mid \langle \varepsilon \rangle \bigwedge_{i \in I} \neg \langle a_i \rangle \top \text{ where } 0 \in I \wedge \alpha_0 = \tau\end{aligned}$$

Weak failures are obtained by intermitting $\langle \varepsilon \rangle$ -observations in between the steps. In comparison to the strong modal characterization of Definition 3.3, this just means prepending productions (and negated actions) by $\langle \varepsilon \rangle$.

Stable failures demand a $\neg \langle \tau \rangle$ -conjunct in the conjunctions of refused actions. In one respect, this increases expressivity, as the possibility to stabilize into a state where $p \not\rightarrow$ becomes observable. In another respect, one loses the option to see failures in instable parts of the transition system. In particular, if all states of a transition system are divergent, stable failure preorder degenerates to weak trace preorder.

Often, weak and stable failures will coincide. For instance, philosopher process P is distinguished from Q by weak failure $\langle \varepsilon \rangle \bigwedge \{ \neg \langle \varepsilon \rangle \langle a \rangle \}$ and by stable failure $\langle \varepsilon \rangle \bigwedge \{ \neg \langle \tau \rangle, \neg \langle a \rangle \}$ alike. For the other direction, Q is preordered to P with respect to both, weak and stable failures.

But in general, weak and stable failure equivalence are *incomparable*. The following example to highlight their differences is due to van Glabbeek:

Example 6.5 (Congruence on τ -abstraction). Figure 6.4 presents transition systems of four processes, given through their initial states: P_e makes a non-deterministic choice op between a and b , performing arbitrarily many idle-actions in between. P_ℓ does the same but can change the choice while idling. P_e^τ and P_ℓ^τ are variants of the two obtained by renaming idle into τ -actions. Many process algebras have an operator to perform such an *abstraction* of a

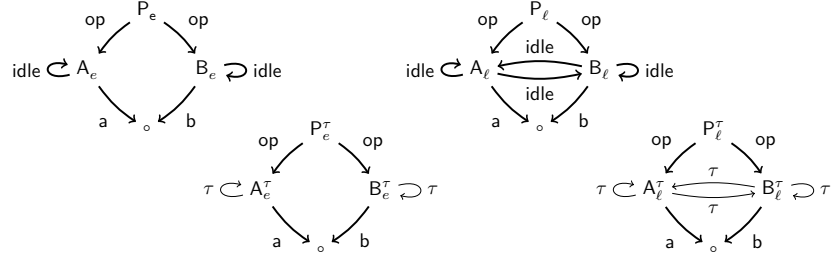


Figure 6.4: A pair of processes P_e and P_l together with versions P_e^τ and P_l^τ of the two where idle has been abstracted into internal τ -behavior.

process, for instance, hide in mCRL2 (Groote & Mousavi, 2014) or hiding “ \backslash ” in CSP (Hoare, 1985).

P_e and P_l allow the same weak failure and stable failure observations and are thus equivalent. For the abstracted variants, P_e^τ is distinguished from P_l^τ by the weak failure $\langle \varepsilon \rangle \langle \text{op} \rangle \langle \varepsilon \rangle \wedge \{ \neg \langle \varepsilon \rangle \langle b \rangle \}$, as b becomes weakly impossible at A_e^τ but not at A_l^τ . Due to the τ -loops, this difference is not expressible as a stable failure. Consequently, P_e^τ and P_l^τ are stable-failure-equivalent.

The example highlights one way in which weak failures might be stronger than stable failures. For most process algebras, weak failures are *too strong* in the following sense: Weak failures cannot be a congruence for hiding operators, as witnessed by Example 6.5!

Therefore, the standard models of CSP following Brookes et al. (1984) have used stable failures for their semantics. Van Glabbeek (1993) only includes variants of stable failures in his weak spectrum, disregarding weak failures. Other authors such as Gazda et al. (2020) work with weak failures. Therefore, our spectrum in Section 6.3 will treat both. Weak failures will be located in the part below weak bisimilarity; stable failures in that below stable bisimilarity.

Remark 6.4 (More on congruences). In general, congruence properties are harder to obtain in the weak spectrum. A classic example would be weak bisimilarity, not forming a congruence with respect to the choice operator $+$, because $a \sim_{\text{WB}} \tau.a$, but $a + b \not\sim_{\text{WB}} \tau.a + b$. We will not go into the particularities around this and point to Sangiorgi (2012, Section 4.4) for how to obtain a congruence through *rooted* weak bisimilarity.

6.3 Expressing the Weak Spectrum by Quantities

We can capture the weak spectrum in a lattice of vectors like the strong one in Chapter 3. But we need a few more dimensions.

6.3.1 Syntactic Expressiveness

We adapt the notions of Section 3.2.2 for the weak spectrum to distinguish the new kinds of conjunctions that appear in HML_{SRBB} of Definition 6.5:

1. Modal depth of *observations* $(\langle a \rangle, (\alpha))$.
2. **New:** Nesting depth of *branching conjunctions* (with one (α) -observation conjunct, not starting with $\langle \varepsilon \rangle$).
3. **New:** Nesting depth of *unstable conjunctions* (that do not enforce stability by a $\neg\langle \tau \rangle \top$ -conjunct).
4. **New:** Nesting depth of *stable conjunctions* (that do enforce stability by a $\neg\langle \tau \rangle \top$ -conjunct).
5. Nesting depth of *immediate conjunctions* (that are not preceded by $\langle \varepsilon \rangle$).
6. Maximal modal depth of *positive conjuncts* in conjunctions.¹⁰⁸
7. Maximal modal depth of *negative conjuncts* in conjunctions.
8. Nesting depth of *negations*.

¹⁰⁸ To simplify matters, we drop the distinction of two kinds of positive conjunct depths of Section 3.2.2. Section 7.3.2 will hint how to add stable revivals back in.


To formalize our weak spectrum, we need to fix the observation languages $\mathcal{O}_N^{\text{weak}}$. This time, we will do this by directly providing an expressiveness pricing metric.

Definition 6.11 (Weak notions). We define the *weak notions of observability* using vectors of extended naturals

$$\mathbf{N}^{\text{weak}} := \mathbb{N}_{\infty}^8,$$

ordered by pointwise comparison of vector components.

We capture the family of *weak observation languages* $\mathcal{O}_{N \in \mathbf{N}^{\text{weak}}}^{\text{weak}}$ by providing an expressiveness price function $\text{expr}^{\text{weak}}: \text{HML}_{\text{SRBB}} \rightarrow \mathbf{N}^{\text{weak}}$ with $\mathcal{O}_N^{\text{weak}} = \{\varphi \mid \text{expr}^{\text{weak}}(\varphi) \leq N\}$.¹⁰⁹ It is defined in mutual recursion with $\text{expr}^{\varepsilon}$ and expr^{\wedge} as follows—if multiple rules apply, pick the first one:

¹⁰⁹  fun Expressiveness_Price .expressiveness_price

$$\begin{aligned}
\text{expr}^{\text{weak}}(\top) &:= \text{expr}^{\varepsilon}(\top) := \mathbf{0} \\
\text{expr}^{\text{weak}}(\langle \varepsilon \rangle \chi) &:= \text{expr}^{\varepsilon}(\chi) \\
\text{expr}^{\text{weak}}(\bigwedge \Psi) &:= \hat{\mathbf{e}}_5 + \text{expr}^{\varepsilon}(\bigwedge \Psi) \\
\text{expr}^{\varepsilon}(\langle a \rangle \varphi) &:= \hat{\mathbf{e}}_1 + \text{expr}^{\text{weak}}(\varphi) \\
\text{expr}^{\varepsilon}(\bigwedge \Psi) &:= \sup \{ \text{expr}^{\wedge}(\psi) \mid \psi \in \Psi \} + \begin{cases} \hat{\mathbf{e}}_4 & \text{if } \neg\langle \tau \rangle \top \in \Psi \\ \hat{\mathbf{e}}_2 + \hat{\mathbf{e}}_3 & \text{if there is } (\alpha)\varphi \in \Psi \\ \hat{\mathbf{e}}_3 & \text{otherwise} \end{cases} \\
\text{expr}^{\wedge}(\neg\langle \tau \rangle \top) &:= (0, 0, 0, 0, 0, 0, 0, 1) \\
\text{expr}^{\wedge}(\neg\varphi) &:= \sup \{ \hat{\mathbf{e}}_8 + \text{expr}^{\text{weak}}(\varphi), \quad (0, 0, 0, 0, 0, 0, (\text{expr}^{\text{weak}}(\varphi))_1, 0) \} \\
\text{expr}^{\wedge}((\alpha)\varphi) &:= \sup \{ \hat{\mathbf{e}}_1 + \text{expr}^{\text{weak}}(\varphi), \quad (0, 0, 0, 0, 0, 1 + (\text{expr}^{\text{weak}}(\varphi))_1, 0, 0) \} \\
\text{expr}^{\wedge}(\varphi) &:= \sup \{ \text{expr}^{\text{weak}}(\varphi), \quad (0, 0, 0, 0, 0, (\text{expr}^{\text{weak}}(\varphi))_1, 0, 0) \}
\end{aligned}$$

6.3.2 Weak Spectrum

Using Definition 6.11, we can give coordinates to the common notions of weak behavioral equivalence in Figure 6.5. In this subsection, we take a deeper look into the observation languages characterized by the coordinates and argue for the correctness of exemplary cases.

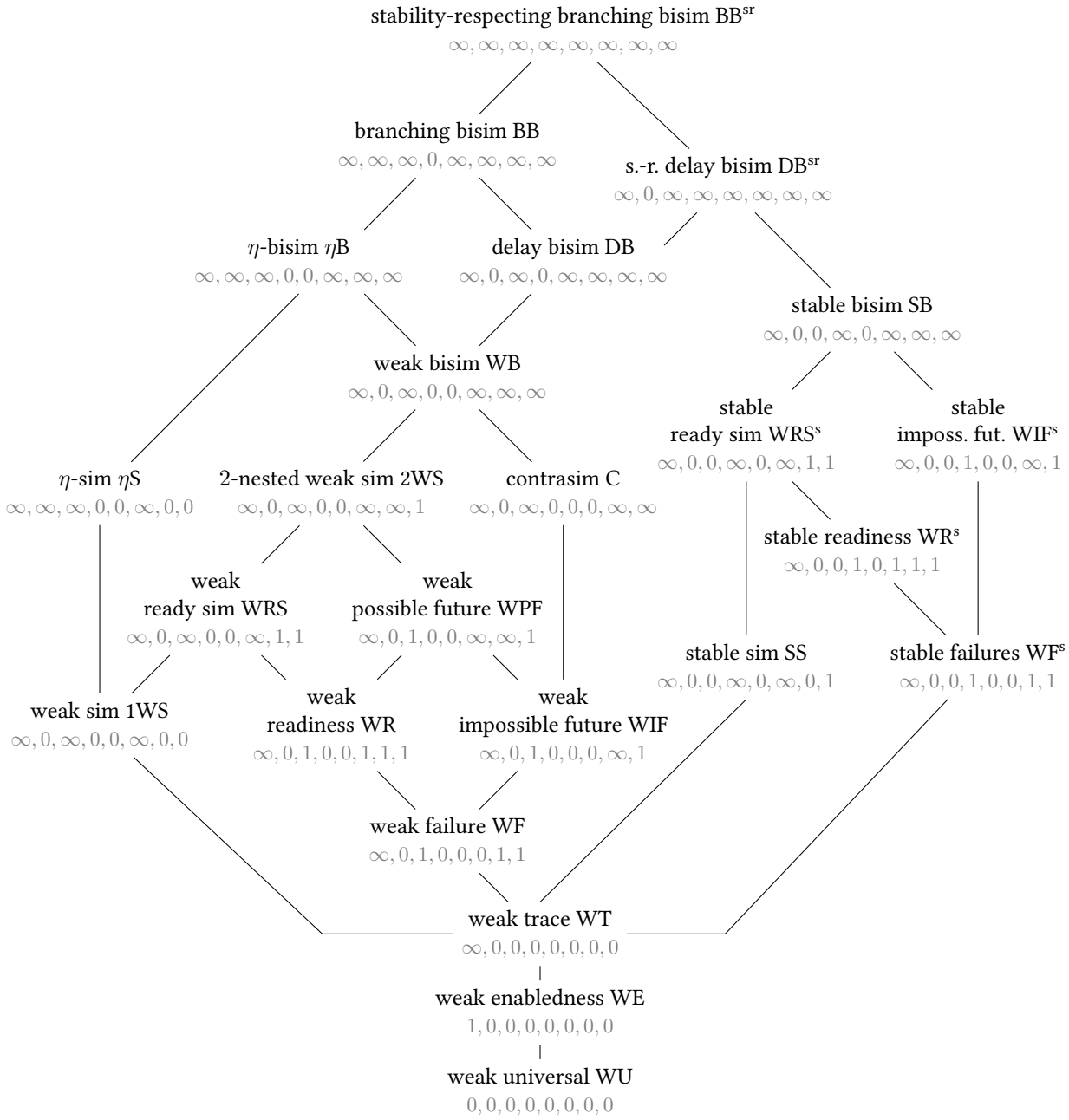


Figure 6.5: Hierarchy of weak behavioral equivalences/preorders, by equivalence notion coordinates.

Linear-time notions. Let us first see how trace-like notions are handled in our spectrum.

Lemma 6.1 (Characterization of weak trace equivalence). *The subset $\mathcal{O}_{\text{WT}}^{\text{weak}} = \mathcal{O}_{(\infty, 0, 0, 0, 0, 0, 0, 0)}^{\text{weak}} \subseteq \text{HML}_{\text{SRBB}}$ can be given by the following grammar:*

$$\varphi_{\text{WT}} ::= \langle \varepsilon \rangle \langle a \rangle \varphi_{\text{WT}} \mid \langle \varepsilon \rangle \top \mid \top \quad \text{with } a \in \text{Act}_{\bullet}.$$

There is a formula $\varphi \in \mathcal{O}_{\text{WT}}^{\text{weak}}$ distinguishing p from q precisely if p is not weakly trace-preordered to q .¹¹⁰

¹¹⁰  lemma [Weak_Traces.LTS_Tau.trace_equals_trace_to_formula](#)

Proof. For a word $\vec{w} = a_1 \dots a_n \in \text{Act}_{\bullet}^*$, the formula $\langle \varepsilon \rangle \langle a_1 \rangle \dots \langle \varepsilon \rangle \langle a_n \rangle \langle \varepsilon \rangle \top \in \mathcal{O}_{\text{WT}}^{\text{weak}}$ is true for a state p precisely if there is a p' such that $p \xrightarrow{\vec{w}} p'$, that is, if $\vec{w} \in \text{WeakTraces}(p)$. As $\text{WeakTraces}(p) = \text{WeakTraces}(q)$ for states p and q precisely if $\text{WeakTraces}(p) \cap \text{Act}_{\bullet}^* = \text{WeakTraces}(q) \cap \text{Act}_{\bullet}^*$, this completes the proof. \square

The whole list of logics for weak linear-time notions can be found in Figure 6.6. The interesting points are highlighted in blue. Productions that could be left out without affecting distinctiveness are set in gray.


The modal characterizations correspond to the ones one would expect, with some trivial ones appearing due to the language hierarchy approach.

Bisimulation-like notions. For the abstractions of bisimilarity in Figure 6.7, some more interesting things happen.

Bisping & Jansen (2025, Section 3.3) prove in detail, for each of our abstractions of bisimilarity, that it corresponds to its respective relational characterization. The Isabelle theory (Barthel et al., 2025) features theorems for η -bisimilarity¹¹¹ and for stability-respecting branching bisimilarity¹¹².

¹¹¹  theorem [Eta_Bisimilarity.LTS_Tau.eta_bisim_coordinate](#)

The first interesting feature of the weak hierarchy is an effect of some languages not enforcing top-level negation as observed in Bisping & Jansen (2024, Ex. 2.5).

¹¹²  theorem [Branching_Bisimilarity.LTS_Tau.sr_branching_bisim_coordinate](#)

Example 6.6 (Weak bisimulation logic). Let us contrast the logic of weak bisimulation observations \mathcal{O}_{WB} defined through $(\infty, 0, \infty, 0, 0, \infty, \infty, \infty)$ to the weak bisimulation observations $\mathcal{O}_{\text{WB}'}$ from Gazda et al. (2020):

$$\varphi_{\text{WB}'} ::= \langle \varepsilon \rangle \varphi_{\text{WB}'} \mid \langle \varepsilon \rangle \langle a \rangle \langle \varepsilon \rangle \varphi_{\text{WB}'} \mid \bigwedge \{ \varphi_{\text{WB}'}, \varphi_{\text{WB}'}, \dots \} \mid \neg \varphi_{\text{WB}'}$$

Our \mathcal{O}_{WB} allows a few formulas that $\mathcal{O}_{\text{WB}'}$ lacks, e.g. $\langle \varepsilon \rangle \langle a \rangle \langle \varepsilon \rangle \langle a \rangle \langle \varepsilon \rangle \top$. This does not add expressiveness as $\mathcal{O}_{\text{WB}'}$ has $\langle \varepsilon \rangle \langle a \rangle \langle \varepsilon \rangle \langle \varepsilon \rangle \langle a \rangle \langle \varepsilon \rangle \top$ and $\llbracket \langle \varepsilon \rangle \langle \varepsilon \rangle \varphi \rrbracket = \llbracket \langle \varepsilon \rangle \varphi \rrbracket$.

For the other direction, there is a bigger difference due to $\mathcal{O}_{\text{WB}'}$ allowing more freedom in the placement of conjunction and negation. In particular, it permits top-level conjunctions and negated conjunctions without $\langle \varepsilon \rangle$ in between. But these features do not add distinctive power. $\mathcal{O}_{\text{WB}'}$ also allows top-level negation, and this adds distinctive power to the preorders, effectively turning them into equivalence relations. We do not enforce this and

Weak universal, WU at $(0, 0, 0, 0, 0, 0, 0, 0)$:

$$\varphi_{\text{WU}} ::= \top \mid \langle \varepsilon \rangle \top$$

Weak enabledness, WE at $(1, 0, 0, 0, 0, 0, 0, 0)$:

$$\varphi_{\text{WE}} ::= \langle \varepsilon \rangle \langle a \rangle \varphi_{\text{WU}} \mid \varphi_{\text{WU}}$$

Weak traces, WT at $(\infty, 0, 0, 0, 0, 0, 0, 0)$:

$$\varphi_{\text{WT}} ::= \langle \varepsilon \rangle \langle a \rangle \varphi_{\text{WT}} \mid \varphi_{\text{WU}}$$

Weak failures, WF at $(\infty, 0, 1, 0, 0, 0, 1, 1)$:

$$\begin{aligned} \varphi_{\text{WF}} &::= \langle \varepsilon \rangle \langle a \rangle \varphi_{\text{WF}} \mid \langle \varepsilon \rangle \bigwedge \{ \psi_{\text{WF}}, \psi_{\text{WF}}, \dots \} \mid \varphi_{\text{WU}} \\ \psi_{\text{WF}} &::= \neg \langle \varepsilon \rangle \langle a \rangle \varphi_{\text{WU}} \mid \neg \langle \varepsilon \rangle \top \mid \langle \varepsilon \rangle \top \end{aligned}$$

Stable failures, WF^s at $(\infty, 0, 0, 1, 0, 0, 1, 1)$:

$$\varphi_{\text{WF}^s} ::= \langle \varepsilon \rangle \langle a \rangle \varphi_{\text{WF}^s} \mid \langle \varepsilon \rangle \bigwedge \{ \neg \langle \tau \rangle \top, \psi_{\text{WF}}, \psi_{\text{WF}}, \dots \} \mid \varphi_{\text{WU}}$$

Weak readiness, WR at $(\infty, 0, 1, 0, 0, 0, 1, 1)$:

$$\begin{aligned} \varphi_{\text{WR}} &::= \langle \varepsilon \rangle \langle a \rangle \varphi_{\text{WR}} \mid \langle \varepsilon \rangle \bigwedge \{ \psi_{\text{WR}}, \psi_{\text{WR}}, \dots \} \mid \varphi_{\text{WU}} \\ \psi_{\text{WR}} &::= \langle \varepsilon \rangle \langle a \rangle \varphi_{\text{WU}} \mid \psi_{\text{WF}} \end{aligned}$$

Stable readiness, WR^s at $(\infty, 0, 0, 1, 0, 1, 1, 1)$:

$$\varphi_{\text{WR}^s} ::= \langle \varepsilon \rangle \langle a \rangle \varphi_{\text{WR}^s} \mid \langle \varepsilon \rangle \bigwedge \{ \neg \langle \tau \rangle \top, \psi_{\text{WR}}, \psi_{\text{WR}}, \dots \} \mid \varphi_{\text{WU}}$$

Weak impossible futures, WIF at $(\infty, 0, 1, 0, 0, 0, \infty, 1)$:

$$\begin{aligned} \varphi_{\text{WIF}} &::= \langle \varepsilon \rangle \langle a \rangle \varphi_{\text{WIF}} \mid \langle \varepsilon \rangle \bigwedge \{ \psi_{\text{WIF}}, \psi_{\text{WIF}}, \dots \} \mid \varphi_{\text{WU}} \\ \psi_{\text{WIF}} &::= \neg \langle \varepsilon \rangle \langle a \rangle \varphi_{\text{WT}} \mid \neg \langle \varepsilon \rangle \top \mid \langle \varepsilon \rangle \top \end{aligned}$$

Stable impossible futures, WIF^s at $(\infty, 0, 0, 1, 0, 0, \infty, 1)$:

$$\varphi_{\text{WIF}^s} ::= \langle \varepsilon \rangle \langle a \rangle \varphi_{\text{WIF}^s} \mid \langle \varepsilon \rangle \bigwedge \{ \neg \langle \tau \rangle \top, \psi_{\text{WIF}}, \psi_{\text{WIF}}, \dots \} \mid \varphi_{\text{WU}}$$

Weak possible futures, WPF at $(\infty, 0, 1, 0, 0, \infty, \infty, 1)$:

$$\begin{aligned} \varphi_{\text{WPF}} &::= \langle \varepsilon \rangle \langle a \rangle \varphi_{\text{WPF}} \mid \langle \varepsilon \rangle \bigwedge \{ \psi_{\text{WPF}}, \psi_{\text{WPF}}, \dots \} \mid \varphi_{\text{WU}} \\ \psi_{\text{WPF}} &::= \langle \varepsilon \rangle \langle a \rangle \varphi_{\text{WT}} \mid \psi_{\text{WIF}} \end{aligned}$$

Figure 6.6: Grammars induced by coordinates for weak linear-time notions of equivalence.

Contrasimulation, C at $(\infty, 0, \infty, 0, 0, 0, \infty, \infty)$:

$$\begin{aligned}\varphi_C &::= \langle \varepsilon \rangle \chi_C \mid \varphi_{WU} \\ \chi_C &::= \langle a \rangle \varphi_C \mid \bigwedge \{ \psi_C, \psi_C, \dots \} \\ \psi_C &::= \neg \langle \varepsilon \rangle \chi_C \mid \langle \varepsilon \rangle \chi_{WU^\times} \\ \chi_{WU^\times} &::= \bigwedge \{ \langle \varepsilon \rangle \chi_{WU^\times}, \neg \langle \varepsilon \rangle \chi_{WU^\times}, \dots \}\end{aligned}$$

Weak bisimulation, WB at $(\infty, 0, \infty, 0, 0, \infty, \infty, \infty)$:

$$\begin{aligned}\varphi_{WB} &::= \langle \varepsilon \rangle \chi_{WB} \mid \varphi_{WU} \\ \chi_{WB} &::= \langle a \rangle \varphi_{WB} \mid \bigwedge \{ \psi_{WB}, \psi_{WB}, \dots \} \\ \psi_{WB} &::= \neg \langle \varepsilon \rangle \chi_{WB} \mid \langle \varepsilon \rangle \chi_{WB}\end{aligned}$$

Delay bisimulation, DB at $(\infty, 0, \infty, 0, \infty, \infty, \infty, \infty)$:

$$\begin{aligned}\varphi_{DB} &::= \langle \varepsilon \rangle \chi_{DB} \mid \bigwedge \{ \psi_{DB}, \psi_{DB}, \dots \} \\ \chi_{DB} &::= \langle a \rangle \varphi_{DB} \mid \bigwedge \{ \psi_{DB}, \psi_{DB}, \dots \} \\ \psi_{DB} &::= \neg \langle \varepsilon \rangle \chi_{DB} \mid \langle \varepsilon \rangle \chi_{DB}\end{aligned}$$

η -bisimulation, ηB at $(\infty, \infty, \infty, 0, 0, \infty, \infty, \infty)$:

$$\begin{aligned}\varphi_{\eta B} &::= \langle \varepsilon \rangle \chi_{\eta B} \mid \varphi_{WU} \\ \chi_{\eta B} &::= \langle a \rangle \varphi_{\eta B} \mid \bigwedge \{ \psi_{\eta B}, \psi_{\eta B}, \dots \} \mid \bigwedge \{ \langle \alpha \rangle \varphi_{\eta B}, \psi_{\eta B}, \psi_{\eta B}, \dots \} \\ \psi_{\eta B} &::= \neg \langle \varepsilon \rangle \chi_{\eta B} \mid \langle \varepsilon \rangle \chi_{\eta B}\end{aligned}$$

Branching bisimulation, BB at $(\infty, \infty, \infty, 0, \infty, \infty, \infty, \infty)$:

$$\begin{aligned}\varphi_{BB} &::= \langle \varepsilon \rangle \chi_{BB} \mid \bigwedge \{ \psi_{BB}, \psi_{BB}, \dots \} \\ \chi_{BB} &::= \langle a \rangle \varphi_{BB} \mid \bigwedge \{ \psi_{BB}, \psi_{BB}, \dots \} \mid \bigwedge \{ \langle \alpha \rangle \varphi_{BB}, \psi_{BB}, \psi_{BB}, \dots \} \\ \psi_{BB} &::= \neg \langle \varepsilon \rangle \chi_{BB} \mid \langle \varepsilon \rangle \chi_{BB}\end{aligned}$$

Stable bisimulation, SB at $(\infty, 0, 0, \infty, 0, 0, \infty, \infty)$:

$$\begin{aligned}\varphi_{SB} &::= \langle \varepsilon \rangle \chi_{SB} \mid \varphi_{WU} \\ \chi_{SB} &::= \langle a \rangle \varphi_{SB} \mid \bigwedge \{ \neg \langle \tau \rangle \top, \psi_{SB}, \psi_{SB}, \dots \} \\ \psi_{SB} &::= \neg \langle \varepsilon \rangle \chi_{SB} \mid \langle \varepsilon \rangle \chi_{SB}\end{aligned}$$

Stability-respecting delay bisimulation, DB^{sr} at $(\infty, 0, \infty, \infty, \infty, \infty, \infty, \infty)$:

$$\begin{aligned}\varphi_{DB^{sr}} &::= \langle \varepsilon \rangle \chi_{DB^{sr}} \mid \bigwedge \{ \psi_{DB^{sr}}, \psi_{DB^{sr}}, \dots \} \\ \chi_{DB^{sr}} &::= \langle a \rangle \varphi_{DB^{sr}} \mid \bigwedge \{ \psi_{DB^{sr}}, \psi_{DB^{sr}}, \dots \} \mid \bigwedge \{ \neg \langle \tau \rangle \top, \psi_{DB^{sr}}, \psi_{DB^{sr}}, \dots \} \\ \psi_{DB^{sr}} &::= \neg \langle \varepsilon \rangle \chi_{DB^{sr}} \mid \langle \varepsilon \rangle \chi_{DB^{sr}}\end{aligned}$$

Stability-respecting branching bisim., BB^{sr} at $(\infty, \infty, \infty, \infty, \infty, \infty, \infty, \infty)$:

$$\begin{aligned}\varphi_{BB^{sr}} &::= \langle \varepsilon \rangle \chi_{BB^{sr}} \mid \bigwedge \{ \psi_{BB^{sr}}, \psi_{BB^{sr}}, \dots \} \\ \chi_{BB^{sr}} &::= \langle a \rangle \varphi_{BB^{sr}} \mid \bigwedge \{ \psi_{BB^{sr}}, \psi_{BB^{sr}}, \dots \} \mid \bigwedge \{ \langle \alpha \rangle \varphi_{BB^{sr}}, \psi_{BB^{sr}}, \psi_{BB^{sr}}, \dots \} \\ &\quad \mid \bigwedge \{ \neg \langle \tau \rangle \top, \psi_{BB^{sr}}, \psi_{BB^{sr}}, \dots \} \\ \psi_{BB^{sr}} &::= \neg \langle \varepsilon \rangle \chi_{BB^{sr}} \mid \langle \varepsilon \rangle \chi_{BB^{sr}}\end{aligned}$$

Figure 6.7: Grammars induced by coordinates for weak bisimulation-like notions of equivalence.

thus our $\preceq_{\mathcal{O}_{\text{WB}}} \neq \sim_{\mathcal{O}_{\text{WB}}}$; for instance, $\tau.a \preceq_{\mathcal{O}_{\text{WB}}} \tau + \tau.a$, but $\tau + \tau.a \not\preceq_{\mathcal{O}_{\text{WB}}} \tau.a$ due to $\langle \varepsilon \rangle \wedge \{\neg \langle \varepsilon \rangle \langle a \rangle \top\}$. However, as a distinction by $\neg \varphi$ in one direction implies one by φ in the other, we know that this difference is ironed out once we consider the equivalence $\sim_{\mathcal{O}_{\text{WB}}}$.

Another point of interest is the relationship of weak bisimilarity, contrasimilarity and stable bisimilarity. Remark 3.3 on synonymous coordinates has mentioned that $(\infty, \infty, 0, 0, \infty, \infty) \in \mathbf{N}^{\text{strong}}$, which disallows positive conjuncts, already characterizes strong bisimilarity. Translating this coordinate to the weak spectrum, we hit $(\infty, 0, \infty, 0, 0, 0, \infty, \infty)$ or $(\infty, 0, 0, \infty, 0, 0, \infty, \infty) \in \mathbf{N}^{\text{weak}}$. In the weak spectrum, these coordinates correspond to differing notions! The first of the two is exactly the coordinate of contrasimilarity. The other one describes observations that are equally expressive to stable bisimulation observations, because stabilized positive conjuncts can be normalized away by ¹¹³

$$\llbracket \langle \varepsilon \rangle \wedge \{\neg \langle \tau \rangle, \varphi_1, \neg \varphi_2, \dots\} \rrbracket = \llbracket \langle \varepsilon \rangle \wedge \{\neg \langle \tau \rangle, \neg \langle \varepsilon \rangle \wedge \{\neg \langle \tau \rangle, \neg \varphi_1\}, \neg \varphi_2, \dots\} \rrbracket.$$

In this sense, contrasimulation and stable bisimilarity are sibling notions in the same manner as weak failures and stable failures of Section 6.2.2.¹¹⁴ We use the higher coordinate $(\infty, 0, 0, \infty, 0, \infty, \infty, \infty)$ with positive conjuncts for stable bisimulation in Figure 6.5 to underscore that stable bisimulation preorder is finer than stable simulation preorder and its variants.


Simulation-like notions. Figure 6.8 adds the simulation-like notions for completeness. At this point, they are not particularly interesting.

Bisping & Jansen (2025) contains proofs that the observation languages for weak simulation and η -simulation correspond to their relational definitions (Definition 6.3).¹¹⁵

Remark 6.5 (Spurious conjuncts and minimality). Several of our weak grammars allow bogus trivial conjuncts. The most severe case might be the trees of ultimately empty conjuncts of χ_{WU^x} for contrasimulation in Figure 6.7. A natural impulse might be to somehow change the grammar to optimize these parts away, ensuring a nicer output of distinguishing formulas in the spectroscopy.

But this optimization would be premature in our context! Spectroscopy strategy formulas will never contain trivial conjuncts, by design, because those can not be distinguishing. Even more, the minimal winning budgets in the spectroscopy game will only ever lead to witness formulas that are minimal with respect to syntactic expressiveness prices. Therefore, optimizing away superfluous conjuncts in the grammars would itself be superfluous.

On the other hand, shrinking the languages would call for additional contexts, that is, non-terminals in the grammar. But more non-terminals mean additional kinds of spectroscopy game positions and moves, thus making the spectroscopy game more complex—for nothing.

¹¹³  theorem [HML_SRBB.LTS_Tau.srb](#)
[_stable_Neg_normalizable_set](#)

¹¹⁴ This is also the topic of Bisping & Montanari (2024, Section 7.3)

¹¹⁵  theorem [Eta_Bisimilarity.LTS_Tau.eta](#)
[_sim_coordinate](#)

Weak simulation, 1WS at $(\infty, 0, \infty, 0, 0, \infty, 0, 0)$:

$$\begin{aligned}\varphi_{1WS} &::= \langle \varepsilon \rangle \langle a \rangle \varphi_{1WS} \mid \langle \varepsilon \rangle \wedge \{\psi_{1WS}, \psi_{1WS}, \dots\} \mid \varphi_{WU} \\ \psi_{1WS} &::= \langle \varepsilon \rangle \langle a \rangle \varphi_{1WS} \mid \langle \varepsilon \rangle \wedge \{\psi_{1WS}, \psi_{1WS}, \dots\}\end{aligned}$$

Weak ready simulation, WRS at $(\infty, 0, \infty, 0, 0, \infty, 1, 1)$:

$$\begin{aligned}\varphi_{WRS} &::= \langle \varepsilon \rangle \langle a \rangle \varphi_{WRS} \mid \langle \varepsilon \rangle \wedge \{\psi_{WRS}, \psi_{WRS}, \dots\} \mid \varphi_{WU} \\ \psi_{WRS} &::= \neg \langle \varepsilon \rangle \langle a \rangle \varphi_{WU^*} \mid \langle \varepsilon \rangle \langle a \rangle \varphi_{WRS} \mid \langle \varepsilon \rangle \wedge \{\psi_{WRS}, \psi_{WRS}, \dots\} \mid \neg \langle \varepsilon \rangle \varphi_{WU^*} \\ \varphi_{WU^*} &::= \top \mid \langle \varepsilon \rangle \wedge \{\varphi_{WU^*}, \varphi_{WU^*}, \dots\}\end{aligned}$$

2-nested weak simulation, 2WS at $(\infty, 0, \infty, 0, 0, \infty, \infty, 1)$:

$$\begin{aligned}\varphi_{2WS} &::= \langle \varepsilon \rangle \langle a \rangle \varphi_{2WS} \mid \langle \varepsilon \rangle \wedge \{\psi_{2WS}, \psi_{2WS}, \dots\} \mid \varphi_{WU} \\ \psi_{2WS} &::= \neg \psi_{1WS} \mid \langle \varepsilon \rangle \langle a \rangle \varphi_{2WS} \mid \langle \varepsilon \rangle \wedge \{\psi_{2WS}, \psi_{2WS}, \dots\}\end{aligned}$$

η -simulation, ηS at $(\infty, \infty, \infty, 0, 0, \infty, 0, 0)$:

$$\begin{aligned}\varphi_{\eta S} &::= \psi_{\eta S} \mid \varphi_{WU} \\ \psi_{\eta S} &::= \langle \varepsilon \rangle \langle a \rangle \varphi_{\eta S} \mid \langle \varepsilon \rangle \wedge \{\langle \alpha \rangle \varphi_{\eta S}, \psi_{\eta S}, \psi_{\eta S}, \dots\} \mid \langle \varepsilon \rangle \wedge \{\psi_{\eta S}, \psi_{\eta S}, \dots\}\end{aligned}$$

Stable simulation, SS at $(\infty, 0, 0, \infty, 0, \infty, 0, 1)$:

$$\begin{aligned}\varphi_{SS} &::= \langle \varepsilon \rangle \langle a \rangle \varphi_{SS} \mid \langle \varepsilon \rangle \wedge \{\neg \langle \tau \rangle \top, \psi_{SS}, \psi_{SS}, \dots\} \mid \varphi_{WU} \\ \psi_{SS} &::= \langle \varepsilon \rangle \langle a \rangle \varphi_{SS} \mid \langle \varepsilon \rangle \wedge \{\neg \langle \tau \rangle \top, \psi_{SS}, \psi_{SS}, \dots\} \mid \neg \varphi_{WU}\end{aligned}$$

Stable ready simulation, WRS^s at $(\infty, 0, 0, \infty, 0, \infty, 1, 1)$:

$$\begin{aligned}\varphi_{WRS^s} &::= \langle \varepsilon \rangle \langle a \rangle \varphi_{WRS^s} \mid \langle \varepsilon \rangle \wedge \{\neg \langle \tau \rangle \top, \psi_{WRS^s}, \psi_{WRS^s}, \dots\} \mid \varphi_{WU} \\ \psi_{WRS^s} &::= \neg \langle \varepsilon \rangle \langle a \rangle \varphi_{WU} \mid \langle \varepsilon \rangle \langle a \rangle \varphi_{WRS^s} \mid \langle \varepsilon \rangle \wedge \{\neg \langle \tau \rangle \top, \psi_{WRS^s}, \psi_{WRS^s}, \dots\}\end{aligned}$$

Figure 6.8: Grammars induced by coordinates for weak simulation-like notions of equivalence.

6.4 Discussion

With this chapter, we have explored the space of weak behavioral equivalences in concurrency theory, focusing on their modal characterizations and relationships within the weak spectrum, following van Glabbeek (1993). This adds the preliminaries to consider their decision procedures in Chapter 7.

The power of weakness. In examples, we have seen how weak equivalences handle internal nondeterminism due to communication in different ways. Practical scenarios, such as parallelizing compilers and process abstraction of Section 6.2, reveal the possibilities and limitations of the different design decisions behind process equivalences.

Following Idea 10, our modal logic characterization through HML_{SRBB} captures a rich weak spectrum between stability-respecting branching bisimilarity and weak trace equivalence. This adds to the rich body of work on *modal characterizations of branching bisimilarity* by De Nicola & Vaandrager (1995), Fokkink et al. (2019), Geuvers (2022), and Geuvers & Golov (2023).

Our spectrum, however, leaves out some classical notions such as stable failure traces and coupled simulation, which would have called for even more dimensions in the already 8-dimensional pricing metric.¹¹⁶ We will return to the topic of how such notions could be included in Section 7.3.2 and 7.3.3.

What's next? Logic and pricing metric are designed in such a way that readers of preceding chapters might already be able to guess how they will translate to game moves for stable and branching conjunctions in the next chapter. Call this avenue for gamification yet another perk of modal characterizations—besides the benefits of built-in transitivity and consistency that we have noticed along the way of this chapter!

¹¹⁶ Especially for coupled similarity, this gap is ironic, as my research into game characterizations of equivalences has been kicked off by an interest in decision procedures for coupled similarity in Bisping & Nestmann (2019) and Bisping et al. (2020). But it would just not have been economic to add a whole new dimension for this single darling of mine.

7 Spectroscopy for the Weak Spectrum

By applying the ideas we have explored so far, we can derive a game for the weak spectrum of Chapter 6. The key new idea here is how to encode *internal activity* “ $\langle \varepsilon \rangle \dots$ ” in game moves:

i Idea 11: Weakening the attacker

In the game, the places where $\langle \varepsilon \rangle$ appears in HML_{SRBB} -formulas mean that the attacker must allow the maximization of defender’s Q -options with respect to internal τ -steps.

Related publications. This chapter is based on “One energy game for the spectrum between branching bisimilarity and weak trace semantics.” (Bisping & Jansen, 2024), its extended draft version (Bisping & Jansen, 2025) and its Isabelle/HOL formalization (Barthel et al., 2025). These publications make the contribution of characterizing the weak spectrum through a *weak spectroscopy* game. The present thesis is more detailed with respect to complexity and to handling of stable failure traces and related notions.

The basic schematics of how the weak game solves the spectroscopy problem are exactly the same as for the strong spectroscopy in Figure 5.1.

What is different this time around is that we have a full *Isabelle/HOL formalization* of game correctness in Section 7.1, and that we can apply the algorithm to real *case studies* that involve internal behavior in Section 7.2. In Section 7.3, we discuss how to use our method to check even *more weak equivalences*.

7.1 The Weak Spectroscopy Game

On the next pages, we *upgrade the spectroscopy game* of Section 5.1 to account for the weak spectrum of Chapter 6.

7.1.1 The Game

The weak spectroscopy game, in many respects, is just like the spectroscopy games we have already discussed: The attacker, trying to distinguish states, has different paths that move closely along the productions of the HML_{SRBB} -grammar (Definition 6.5). But this time, there are *four* different kinds of non-empty conjunctions! This makes the following schematic depiction in Figure 7.1 already look quite entangled.

Formally, the game rules are defined as follows:

Definition 7.1 (Weak spectroscopy game). For a system $\mathcal{S} = (\mathcal{P}, \text{Act}, \rightarrow)$, the 8-dimensional *weak spectroscopy game* $\mathcal{G}_{\nabla}^{\mathcal{S}} = (G, G_{\text{a}}, \succrightarrow, w)$ consists of

- *attacker (immediate) positions* $[p, Q]_{\text{a}} \in G_{\text{a}}$,

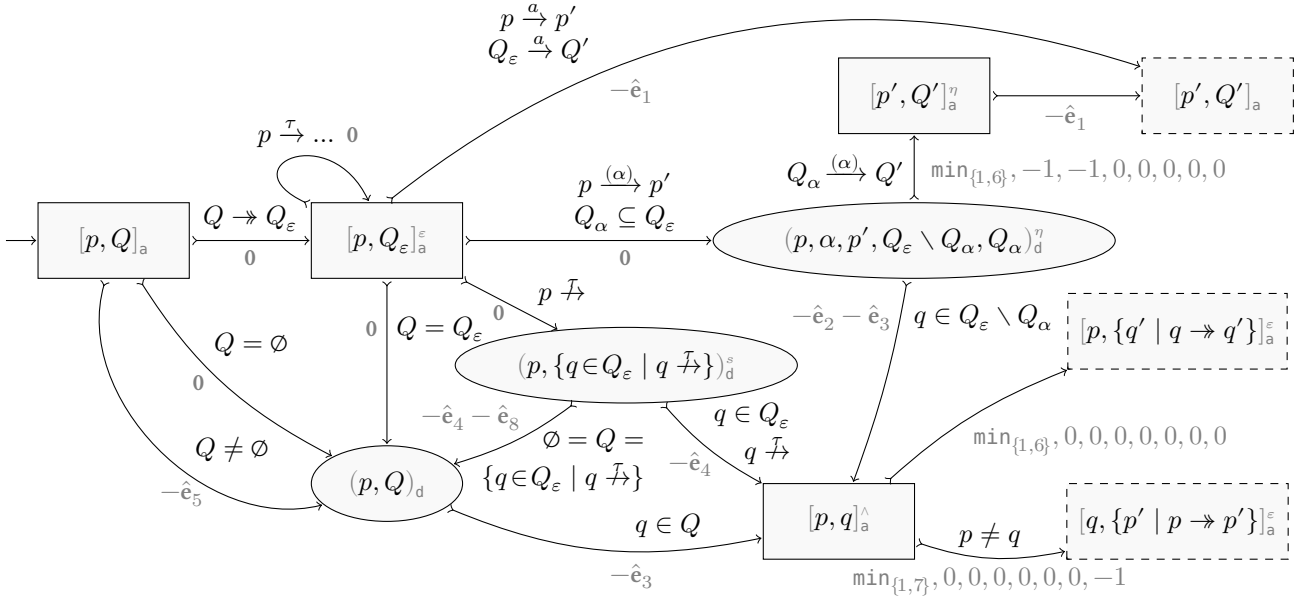


Figure 7.1: Schematic weak spectroscopy game of Definition 7.1.

- **new:** attacker delayed positions $[p, Q]_a^\varepsilon \in G_a$,
- **attacker conjunct positions** $[p, q]_a^\wedge \in G_a$,
- **new:** attacker branching positions $[p, Q]_a^\eta \in G_d$,
- **defender conjunction positions** $(p, Q)_d \in G_d$,
- **new:** defender stable conjunction positions $(p, Q)_d^s \in G_d$,
- **new:** defender branching positions $(p, \alpha, p', Q, Q_\alpha)_d^\eta \in G_d$,

¹¹⁷ inductive [Spectroscopy_Game.LTS_Tau](#)
[.spectroscopy_moves](#)

where $p, q \in \mathcal{P}$ and $Q, Q_\alpha \in 2^{\mathcal{P}}$, and the following sixteen kinds of moves.¹¹⁷

Moves to allow *internal behavior* “ $\langle \varepsilon \rangle \dots$ ” between observations:

$$\begin{array}{ll} \text{delay} & [p, Q]_a \xrightarrow{0,0,0,0,0,0,0,0} [p, Q]_a^\varepsilon \quad \text{if } Q \twoheadrightarrow Q', \\ \text{procrastination} & [p, Q]_a \xrightarrow{0,0,0,0,0,0,0,0} [p', Q]_a^\varepsilon \quad \text{if } p \xrightarrow{\tau} p', p \neq p', \end{array}$$

moves to represent the *known HML constructs* “ $\langle a \rangle \dots$ ”, “ $\wedge \{ \dots \}$ ”, and “ $\neg \dots$ ” in the now *two* contexts:

$$\begin{array}{lll} \text{observation} & [p, Q]_a^\varepsilon \xrightarrow{-1,0,0,0,0,0,0,0} [p', Q']_a^\varepsilon & \text{if } p \xrightarrow{a} p', Q \xrightarrow{a} Q', a \neq \tau, \\ \text{finishing} & [p, \emptyset]_a \xrightarrow{0,0,0,0,0,0,0,0} (p, \emptyset)_d & \\ \text{immediate conj.} & [p, Q]_a \xrightarrow{0,0,0,0,-1,0,0,0} (p, Q)_d & \text{if } Q \neq \emptyset, \\ \text{late conj.} & [p, Q]_a^\varepsilon \xrightarrow{0,0,0,0,0,0,0,0} (p, Q)_d, & \\ \text{conj. answer} & (p, Q)_d \xrightarrow{0,0,-1,0,0,0,0,0} [p, q]_a^\wedge & \text{if } q \in Q, \\ \text{positive conjunct} & [p, q]_a^\wedge \xrightarrow{\min\{1,6\},0,0,0,0,0,0,0} [p, Q]_a^\varepsilon & \text{if } \{q\} \twoheadrightarrow Q, \\ \text{negative conjunct} & [p, q]_a^\wedge \xrightarrow{\min\{1,7\},0,0,0,0,0,0,-1} [q, Q]_a^\varepsilon & \text{if } \{p\} \twoheadrightarrow Q \text{ and } p \neq q, \end{array}$$

moves to encode *stable conjunctions* “ $\bigwedge\{\neg\langle\tau\rangle\top, \psi, \psi, \dots\}$ ”:

$$\begin{array}{lll} \text{stable conj.} & [p, Q]_a^\varepsilon & \xrightarrow{0,0,0,0,0,0,0,0} (p, Q')_d^s \quad \text{if } Q' = \{q \in Q \mid q \not\sim\}, p \not\sim, \\ \text{stable conj. answer} & (p, Q)_d^s & \xrightarrow{0,0,0,-1,0,0,0,0} [p, q]_a^\wedge \quad \text{if } q \in Q, \\ \text{stable finishing} & (p, \emptyset)_d^s & \xrightarrow{0,0,0,-1,0,0,0,-1} (p, \emptyset)_d, \end{array}$$

and moves to encode *branching conjunctions* “ $\bigwedge\{(\alpha)\varphi, \psi, \psi, \dots\}$ ”:

$$\begin{array}{lll} \text{branching conj.} & [p, Q]_a^\varepsilon & \xrightarrow{0,0,0,0,0,0,0,0} (p, \alpha, p', Q \setminus Q_\alpha, Q_\alpha)_d^\eta \quad \text{if } p \xrightarrow{(\alpha)} p', Q_\alpha \subseteq Q, \\ \text{branch. answer} & (p, \alpha, p', Q, Q_\alpha)_d^\eta & \xrightarrow{0,-1,-1,0,0,0,0,0} [p, q]_a^\wedge \quad \text{if } q \in Q, \\ \text{branch. observation} & (p, \alpha, p', Q, Q_\alpha)_d^\eta & \xrightarrow{\min\{1,6\}, -1,-1,0,0,0,0,0} [p', Q']_a^\eta \quad \text{with } Q_\alpha \xrightarrow{(\alpha)} Q', \\ \text{branch. accounting} & [p, Q]_a^\eta & \xrightarrow{-1,0,0,0,0,0,0,0} [p, Q]_a. \end{array}$$

Intuitively, the attacker is heavily weakened in the weak spectroscopy game due to the requirement to pass delay-moves in order to formulate observation attacks. Following Idea 11, these moves grow the right-hand-sets, which increases defender options. At the same time, there are several new tactical possibilities for the attacker that correspond to the special weak conjunctions, as can be seen in the following example.

Example 7.1 (Failures of philosophy). For our standard philosopher system of Figure 2.3, Example 6.2 has determined that P and Q are not weakly bisimilar, but weakly similar (and thus weakly trace-equivalent).

The stable failure $\langle\varepsilon\rangle \bigwedge\{\neg\langle\varepsilon\rangle\langle a\rangle\}$ discussed in Section 6.2.2 corresponds to the following game moves, which need budget $(1, 0, 0, 1, 0, 0, 1, 1)$:

$$\begin{array}{lll} [P, \{Q\}]_a & \xrightarrow{0} [P, \{Q, q_{ab}\}]_a^\varepsilon & \text{(delay)} \\ & \xrightarrow{0} [p_b, \{Q, q_{ab}\}]_a^\varepsilon & \text{(procrastination)} \\ & \xrightarrow{0} (p_b, \{q_{ab}\})_d & \text{(stable conj.)} \\ & \xrightarrow{-\varepsilon_4} [p_b, q_{ab}]_a^\wedge & \text{(stable conj. answer)} \\ \xrightarrow{\min\{1,7\}, 0,0,0,0,0,0,-1} & [q_{ab}, \{p_b\}]_a^\varepsilon & \text{(negative conjunct)} \\ & \xrightarrow{-\varepsilon_1} [q_1, \emptyset]_a & \text{(a-observation)} \\ & \xrightarrow{0} (q_1, \emptyset)_d \not\sim & \text{(finishing)} \end{array}$$

A similar sequence of moves works for weak failures. For weak simulation, we know that P is strongly simulated by Q, which transfers to the weak game in the sense that we also cannot find a weak simulation distinction in this direction. The attacker must pass through a negative-conjunct move to exploit that P resolves the choice more quickly than Q.

For the other direction, Example 6.2 has established that weak simulation, $Q \preceq_{\text{WS}} P$, holds as well. But how does this go together with Example 5.3 mentioning that strong simulation is disproved by the HML formula $\langle\tau\rangle \bigwedge\{\langle a\rangle, \langle b\rangle\}$? In the weak game, there are no strong τ -observation moves. The closest weak equivalent is to just use delay/procrastination, corresponding to the HML_{SRBB} formula $\langle\varepsilon\rangle \bigwedge\{\langle\varepsilon\rangle\langle a\rangle, \langle\varepsilon\rangle\langle b\rangle\}$. But this fails to be a distinction in the weak game as the defender can just stay at $P \in \{P, p_a, p_b\}$.

However, other forms of abstracted simulation can be used to force the defender out of P without negations:

- *η -simulation* allows branching conjunction moves: $[q_{ab}, \{P, p_a, p_b\}]_a^e \xrightarrow{\circ} (q_{ab}, a, q_1, \{p_a\}, \{P, p_b\})_d^\eta$. After this, p_a is discharged through a positive conjunct b-observation, possible from q_{ab} ; and the $\{P, p_b\}$ -option directly ends in $\xrightarrow{\circ} [q_1, \emptyset]_a^\eta \xrightarrow{\circ} (q_1, \emptyset)_d \not\models$, as neither P nor p_b allow immediate \xrightarrow{a} -steps. These moves cost $(1, 1, 1, 0, 0, 1, 0, 0)$ and correspond to the distinguishing formula $\langle \varepsilon \rangle \wedge \{ \langle a \rangle, \langle \varepsilon \rangle \langle b \rangle \}$.
- *Stable simulation* allows stable conjunction moves: $[q_{ab}, \{P, p_a, p_b\}]_a^e \xrightarrow{\circ} (q_{ab}, \{P', p_a, p_b\})_d^s$. After this, q_{ab} can out-maneuver p_a through positive conjunct b-observation, and p_b through positive conjunct a-observation. The moves need $(1, 0, 1, 1, 0, 1, 0, 1)$ energy and match the distinguishing formula $\langle \varepsilon \rangle \wedge \{ \neg \langle \tau \rangle, \langle \varepsilon \rangle \langle a \rangle, \langle \varepsilon \rangle \langle b \rangle \}$.

In summary, the game moves show that P and Q are weakly similar, but no notion besides or above in the weak spectrum (Figure 6.5) can hold because weak failures, stable failures, η -simulation, and stable simulation can be disproven by attacker moves for at least one direction. This reasoning, of course, depends on the weak spectroscopy game being correct ...

7.1.2 Correctness

Establishing correctness now proceeds mostly as we are used to from Section 4.2.3 and 5.1.2. This subsection limits itself to citing the lemma heads and the inductive predicates used in the full proofs of Bisping & Jansen (2025).

The following facts moreover have been fully formalized in Isabelle/HOL, which will be the topic of the following Section 7.1.3.

Definition 7.2 (Strategy formulas for \mathcal{G}_∇). The set of *strategy formulas* for a game position g and a budget e , $\text{Strat}_\nabla(g, e)$, in the context of a weak spectroscopy game \mathcal{G}_∇^S is defined inductively by the rules in Figure 7.2.¹¹⁸


Attacker winning budgets can be translated into strategy formulas of matching price, as illustrated in Example 7.1.

Lemma 7.1 (Distinction formulatability). *If $e \in \text{Win}_a^{\mathcal{G}_\nabla}([p, Q]_a)$, then there is $\varphi \in \text{Strat}_\nabla([p, Q]_a, e)$ with $\text{expr}^{\text{weak}}(\varphi) \leq e$.*¹¹⁹

Approach. By induction over game positions g and energies e according to the inductive characterization of attacker winning budgets Proposition 4.3 and with respect to the following property:

1. If $e \in \text{Win}_a([p, Q]_a)$, then there is $\varphi \in \text{Strat}_\nabla([p, Q]_a, e)$ with $\text{expr}^{\text{weak}}(\varphi) \leq e$;
2. If $e \in \text{Win}_a([p, Q]_a^\varepsilon)$, then there is $\chi \in \text{Strat}_\nabla([p, Q]_a^\varepsilon, e)$ with $\text{expr}^\varepsilon(\chi) \leq e$;
3. If $e \in \text{Win}_a([p, q]_a^\wedge)$, then there is $\psi \in \text{Strat}_\nabla([p, q]_a^\wedge, e)$ with $\text{expr}^\wedge(\psi) \leq e$;

¹¹⁸  inductive [Strategy_Formulas.weak_spectroscopy_game.strategy_formula](#)

¹¹⁹  lemma [Strategy_Formulas.weak_spectroscopy_game.winning_budget_implies_strategy_formula](#)

$$\begin{array}{l}
\text{delay} \frac{[p, Q]_a \xrightarrow{u} [p, Q']_a \quad e' = \text{upd}(u, e) \in \text{Win}_a([p, Q']_a) \quad \chi \in \text{Strat}_\nabla([p, Q']_a, e')}{\langle \varepsilon \rangle \chi \in \text{Strat}_\nabla([p, Q]_a, e)} \\
\\
\text{procrastination} \frac{[p, Q]_a \xrightarrow{u} [p', Q']_a \quad e' = \text{upd}(u, e) \in \text{Win}_a([p', Q']_a) \quad \chi \in \text{Strat}_\nabla([p', Q']_a, e')}{\chi \in \text{Strat}_\nabla([p, Q]_a, e)} \\
\\
\text{observation} \frac{\begin{array}{c} [p, Q]_a \xrightarrow{u} [p', Q']_a \quad e' = \text{upd}(u, e) \in \text{Win}_a([p', Q']_a) \\ p \xrightarrow{a} p' \quad Q \xrightarrow{a} Q' \quad \varphi \in \text{Strat}_\nabla([p', Q']_a, e') \end{array}}{\langle a \rangle \varphi \in \text{Strat}_\nabla([p, Q]_a, e)} \\
\\
\text{immediate conj.} \frac{[p, Q]_a \xrightarrow{u} (p, Q)_d \quad e' = \text{upd}(u, e) \in \text{Win}_a((p, Q)_d) \quad \varphi \in \text{Strat}_\nabla((p, Q)_d, e')}{\varphi \in \text{Strat}_\nabla([p, Q]_a, e)} \\
\\
\text{late conj.} \frac{[p, Q]_a \xrightarrow{u} (p, Q)_d \quad e' = \text{upd}(u, e) \in \text{Win}_a((p, Q)_d) \quad \chi \in \text{Strat}_\nabla((p, Q)_d, e')}{\chi \in \text{Strat}_\nabla([p, Q]_a, e)} \\
\\
\text{conj. answer} \frac{\begin{array}{c} (p, Q)_d \xrightarrow{u_q} [p, q]_a^\wedge \\ \forall q \in Q. \quad e_q = \text{upd}(u_q, e) \in \text{Win}_a([p, q]_a^\wedge) \wedge \psi_q \in \text{Strat}_\nabla([p, q]_a^\wedge, e_q) \end{array}}{\bigwedge \{ \psi_q \mid q \in Q \} \in \text{Strat}_\nabla((p, Q)_d, e)} \\
\\
\text{positive conjunct} \frac{[p, q]_a^\wedge \xrightarrow{u} [p, Q']_a \quad e' = \text{upd}(u, e) \in \text{Win}_a([p, Q']_a) \quad \chi \in \text{Strat}_\nabla([p, Q']_a, e')}{\langle \varepsilon \rangle \chi \in \text{Strat}_\nabla([p, q]_a^\wedge, e)} \\
\\
\text{negative conjunct} \frac{[p, q]_a^\wedge \xrightarrow{u} [q, P']_a \quad e' = \text{upd}(u, e) \in \text{Win}_a([q, P']_a) \quad \chi \in \text{Strat}_\nabla([q, P']_a, e')}{\neg \langle \varepsilon \rangle \chi \in \text{Strat}_\nabla([p, q]_a^\wedge, e)} \\
\\
\text{stable conj.} \frac{[p, Q]_a \xrightarrow{u} (p, Q')_d^\dagger \quad e' = \text{upd}(u, e) \in \text{Win}_a((p, Q')_d^\dagger) \quad \chi \in \text{Strat}_\nabla((p, Q')_d^\dagger, e')}{\chi \in \text{Strat}_\nabla([p, Q]_a, e)} \\
\\
\text{stable conj. answer} \frac{\begin{array}{c} (p, Q)_d^\dagger \xrightarrow{u_q} [p, q]_a^\wedge \quad Q \neq \emptyset \\ \forall q \in Q. \quad e_q = \text{upd}(u_q, e) \in \text{Win}_a([p, q]_a^\wedge) \wedge \psi_q \in \text{Strat}_\nabla([p, q]_a^\wedge, e_q) \end{array}}{\bigwedge (\{ \neg \langle \tau \rangle \top \} \cup \{ \psi_q \mid q \in Q \}) \in \text{Strat}_\nabla((p, Q)_d^\dagger, e)} \\
\\
\text{stable finishing} \frac{(p, \emptyset)_d^\dagger \xrightarrow{u} (p, \emptyset)_d \quad e' = \text{upd}(u, e) \in \text{Win}_a((p, \emptyset)_d)}{\bigwedge \{ \neg \langle \tau \rangle \top \} \in \text{Strat}_\nabla((p, Q)_d^\dagger, e)} \\
\\
\text{branching conj.} \frac{\begin{array}{c} [p, Q]_a \xrightarrow{u} (p, \alpha, p', Q', Q_\alpha)_d^\eta \quad e' = \text{upd}(u, e) \in \text{Win}_a((p, \alpha, p', Q', Q_\alpha)_d^\eta) \\ \chi \in \text{Strat}_\nabla((p, \alpha, p', Q', Q_\alpha)_d^\eta, e') \end{array}}{\chi \in \text{Strat}_\nabla([p, Q]_a, e)} \\
\\
\text{branch. answer} \frac{\begin{array}{c} g_d = (p, \alpha, p', Q, Q_\alpha)_d^\eta \xrightarrow{u_\alpha} [p', Q']_a^\eta \xrightarrow{u'_\alpha} [p', Q']_a \\ e_\alpha = \text{upd}(u'_\alpha, \text{upd}(u_\alpha, e)) \in \text{Win}_a([p', Q']_a) \quad \varphi_\alpha \in \text{Strat}_\nabla([p', Q']_a, e_\alpha) \\ \forall q \in Q. \quad g_d \xrightarrow{u_q} [p, q]_a^\wedge \wedge e_q = \text{upd}(u_q, e) \in \text{Win}_a([p, q]_a^\wedge) \wedge \psi_q \in \text{Strat}_\nabla([p, q]_a^\wedge, e_q) \end{array}}{\bigwedge (\{ (\alpha) \varphi_\alpha \} \cup \{ \psi_q \mid q \in Q \}) \in \text{Strat}_\nabla((p, \alpha, p', Q, Q_\alpha)_d^\eta, e)}
\end{array}$$

Figure 7.2: Strategy formulas for the weak spectroscopy.

4. If $e \in \text{Win}_a((p, Q)_d)$, then there is $\bigwedge \Psi \in \text{Strat}_\nabla((p, Q)_d, e)$ with $\text{expr}^\varepsilon(\bigwedge \Psi) \leq e$;
5. If $e \in \text{Win}_a((p, Q)_d^s)$, then there is $\bigwedge (\{\neg\langle\tau\rangle\top\} \cup \Psi) \in \text{Strat}_\nabla((p, Q)_d^s, e)$ with $\text{expr}^\varepsilon(\bigwedge (\{\neg\langle\tau\rangle\top\} \cup \Psi)) \leq e$;
6. If $e \in \text{Win}_a((p, \alpha, p', Q \setminus Q_\alpha, Q_\alpha)_d^s)$, then there is $\bigwedge (\{(\alpha)\varphi'\} \cup \Psi) \in \text{Strat}_\nabla((p, \alpha, p', Q \setminus Q_\alpha, Q_\alpha)_d^s, e)$ with $\text{expr}^\varepsilon(\bigwedge (\{(\alpha)\varphi'\} \cup \Psi)) \leq e$.

□

Weak strategy formulas distinguish the left state from the set of states on the right, where we lift Definition 2.13 to sets as follows:

Definition 7.3 (Distinguishes from set). We say a formula φ distinguishes a state $p \in \mathcal{P}$ from a set of states $Q \subseteq \mathcal{P}$ iff $p \in \llbracket \varphi \rrbracket$ and $Q \cap \llbracket \varphi \rrbracket = \emptyset$.

Lemma 7.2 (Distinction soundness). If $\varphi \in \text{Strat}_\nabla([p, Q]_a, e)$, then φ distinguishes p from Q .¹²⁰

Approach. By induction over the derivation of $\dots \in \text{Strat}_\nabla(g, e)$ according to Definition 7.2 on the following inductive property:

1. If $\varphi \in \text{Strat}_\nabla([p, Q]_a, e)$, then φ distinguishes p from Q ;
2. If $\chi \in \text{Strat}_\nabla([p, Q]_a^s, e)$ and $Q \twoheadrightarrow Q$, then $\langle \varepsilon \rangle \chi$ distinguishes p from Q ;
3. If $\psi \in \text{Strat}_\nabla([p, q]_a^\wedge, e)$, then ψ distinguishes p from $\{q\}$;
4. If $\bigwedge \Psi \in \text{Strat}_\nabla((p, Q)_d, e)$, then $\bigwedge \Psi$ distinguishes p from Q ;
5. If $\bigwedge (\{\neg\langle\tau\rangle\top\} \cup \Psi) \in \text{Strat}_\nabla((p, Q)_d^s, e)$ and p is stable, then the stable conjunction $\bigwedge (\{\neg\langle\tau\rangle\top\} \cup \Psi)$ distinguishes p from Q ;
6. If $\bigwedge (\{(\alpha)\varphi'\} \cup \Psi) \in \text{Strat}_\nabla((p, \alpha, p', Q \setminus Q_\alpha, Q_\alpha)_d^s, e)$, $p \xrightarrow{(\alpha)} p'$ and $Q_\alpha \subseteq Q$, then the branching conjunction $\bigwedge (\{(\alpha)\varphi'\} \cup \Psi)$ distinguishes p from Q .

□


Distinguishing formulas certify the existence of equally cheap ways for the attacker to win.

Lemma 7.3 (Distinction completeness). If $\varphi \in \text{HML}_{\text{SRBB}}$ distinguishes p from Q , then $\text{expr}^{\text{weak}}(\varphi) \in \text{Win}_a^{\mathcal{G}_\nabla}([p, Q]_a)$.¹²¹

Approach. By mutual structural induction on φ , χ , and ψ with respect to the following claims:

1. If $\varphi \in \text{HML}_{\text{SRBB}}$ distinguishes p from $Q \neq \emptyset$, then $\text{expr}^{\text{weak}}(\varphi) \in \text{Win}_a([p, Q]_a)$;
2. If χ distinguishes p from $Q \neq \emptyset$ and Q is closed under \twoheadrightarrow (that is $Q \twoheadrightarrow Q$), then $\text{expr}^\varepsilon(\chi) \in \text{Win}_a([p, Q]_a^s)$;
3. If ψ distinguishes p from q , then $\text{expr}^\wedge(\psi) \in \text{Win}_a([p, q]_a^\wedge)$;
4. If $\bigwedge \Psi$ distinguishes p from $Q \neq \emptyset$, then $\text{expr}^\varepsilon(\bigwedge \Psi) \in \text{Win}_a((p, Q)_d)$;
5. If $\bigwedge \{\neg\langle\tau\rangle\top\} \cup \Psi$ distinguishes p from $Q \neq \emptyset$ and the processes in Q are stable, then $\text{expr}^\varepsilon(\bigwedge \{\neg\langle\tau\rangle\top\} \cup \Psi) \in \text{Win}_a((p, Q)_d^s)$;

¹²⁰  lemma Strategy_Formulas.weak_spectroscopy_game.strategy_formulas_distinguish

¹²¹  lemma Distinction_Implies_Winning_Budgets.weak_spectroscopy_game.distinction_implies_winning_budgets

6. If $\bigwedge\{(\alpha)\varphi'\} \cup \Psi$ distinguishes p from Q , then, for any $p \xrightarrow{(\alpha)} p' \in \llbracket \varphi' \rrbracket$ and $Q_\alpha = Q \setminus \llbracket (\alpha)\varphi' \rrbracket$, $\text{expr}^\varepsilon(\bigwedge\{(\alpha)\varphi'\} \cup \Psi) \in \text{Win}_a((p, \alpha, p', Q \setminus Q_\alpha, Q_\alpha)_d^\eta)$. \square

Theorem 7.1 (\mathbf{N}^{weak} -characterization). *For all $N \in \mathbf{N}^{\text{weak}}$, $p \in \mathcal{P}$, $Q \in 2^{\mathcal{P}}$, the following are equivalent:*¹²²

- There exists a formula $\varphi \in \text{HML}_{\text{SRBB}}$ with price $\text{expr}^{\text{weak}}(\varphi) \leq N$ that distinguishes p from Q .
- Attacker wins \mathcal{G}_{∇}^S from $[p, Q]_a$ with energy N .

¹²²  theorem [Silent_Step_Spectroscopy.weak_spectroscopy_game.spectroscopy_game_correctness](#)


7.1.3 Isabelle/HOL Formalization

Barthel et al. (2025) formalize the correctness result for the weak spectroscopy game in the interactive proof assistant Isabelle/HOL. The preceding definitions and facts have already linked to their respective Isabelle/HOL counterparts. This subsection is devoted to providing some insights into the formalization.¹²³ We take a tour through roughly a hundred of the most interesting lines of the 6500 line theory development.

The weak spectroscopy game (Definition 7.1) is modelled through a *parametric datatype* ('s, 'a) spectroscopy_position for its positions and a *partial function* spectroscopy_moves to determine the moves connecting them.¹²⁴ The parameter types 's and 'a capture the states \mathcal{P} and actions Act of the transition system on which we operate.¹²⁵

¹²³ For questions on the Isabelle/Isar language, consult *The Isabelle/Isar reference manual* (Wenzel, 2025).


¹²⁴ To be precise: “Partial function” here means that it returns option-values, which might either be Some x with an output or None otherwise. All functions in higher-order logic are total.

¹²⁵  datatype [Spectroscopy_Game.spectroscopy_position](#)

```
datatype ('s, 'a) spectroscopy_position =
  Attacker_Immediate
    (attacker_state: <'s>) (defender_states: <'s set>)
| Attacker_Delayed
    (attacker_state: <'s>) (defender_states: <'s set>)
| ...
| Defender_Branch
    (attacker_state: <'s>) (attack_action: <'a>)
    (attacker_state_succ: <'s>)
    (defender_states: <'s set>)
    (defender_branch_states: <'s set>)

fun spectroscopy_moves (in LTS_Tau) ::
  <('s, 'a) spectroscopy_position => ('s, 'a) spectroscopy_position
  => energy update option>
where
  delay: <spectroscopy_moves
    (Attacker_Immediate p Q) (Attacker_Delayed p' Q')
  = (if p' = p ^ Q =>S Q' then id_up else None)>
| [...]
```

The game itself is then built as a combination of the *locale* for transition systems with internal actions LTS_Tau (with a transition relation step) and an energy_game locale. The latter is instantiated with the moves, a predicate spectroscopy_defender singling out defender positions, and the \leq -relation on 8-dimensional energies.¹²⁶

¹²⁶  locale Spectroscopy_Game.weak_spectroscopy_game

```
locale weak_spectroscopy_game =
  LTS_Tau step  $\tau$ 
  + energy_game <spectroscopy_moves> <spectroscopy_defender> <( $\leq$ )>
  for step :: <'s  $\Rightarrow$  'a  $\Rightarrow$  's  $\Rightarrow$  bool> (<_  $\mapsto$  _> [70, 70, 70] 80)
  and  $\tau$  :: 'a
```

Within the locale, we can establish our correctness results.

The strategy formulas Strat_∇ appear as three mutually *inductive predicates*, because the grammar of HML_{SRBB} (Definition 6.5) is implemented as three mutually recursive data types (one per non-terminal).¹²⁷

¹²⁷  inductive Strategy_Formulas.weak_spectroscopy_game.strategy_formula

```
inductive
  strategy_formula :: <('s, 'a) spectroscopy_position
     $\Rightarrow$  energy  $\Rightarrow$  ('a, 's) hml_srbb  $\Rightarrow$  bool>
and
  strategy_formula_inner :: <('s, 'a) spectroscopy_position
     $\Rightarrow$  energy  $\Rightarrow$  ('a, 's) hml_srbb_inner  $\Rightarrow$  bool>
and
  strategy_formula_conjunct :: <('s, 'a) spectroscopy_position
     $\Rightarrow$  energy  $\Rightarrow$  ('a, 's) hml_srbb_conjunct  $\Rightarrow$  bool>
where
  delay: <strategy_formula (Attacker_Immediate p Q) e (Internal  $\chi$ )>
  if < $\exists Q'$ . spectroscopy_moves
    (Attacker_Immediate p Q) (Attacker_Delayed p Q') = id_up
     $\wedge$  attacker_wins e (Attacker_Delayed p Q')
     $\wedge$  strategy_formula_inner (Attacker_Delayed p Q') e  $\chi$ >
  | [...]
```

We then reproduce the induction of Lemma 7.1 on attacker winning budgets in the following lemma. For this, the theory uses the inductive characterization of Win_a in Proposition 4.3 as definition for attacker_wins.¹²⁸

¹²⁸  lemma Strategy_Formulas.weak_spectroscopy_game.winning_budget_implies_strategy_formula

```
lemma winning_budget_implies_strategy_formula:
  assumes <attacker_wins e g>
  shows
    <case g of
      Attacker_Immediate p Q  $\Rightarrow$ 
         $\exists \varphi$ . strategy_formula g e  $\varphi \wedge$  expressiveness_price  $\varphi \leq e$ 
    | Attacker_Delayed p Q  $\Rightarrow$ 
         $\exists \chi$ . strategy_formula_inner g e  $\chi \wedge$  expr_pr_inner  $\chi \leq e$ 
```

```

| Attacker_Conjunct p q ⇒
  ∃ψ. strategy_formula_conjunct g e ψ
    ∧ expr_pr_conjunct ψ ≤ e
| Defender_Conj p Q ⇒
  ∃χ. strategy_formula_inner g e χ ∧ expr_pr_inner χ ≤ e
| Defender_Stable_Conj p Q ⇒
  ∃χ. strategy_formula_inner g e χ ∧ expr_pr_inner χ ≤ e
| Defender_Branch p α p' Q Qa ⇒
  ∃χ. strategy_formula_inner g e χ ∧ expr_pr_inner χ ≤ e
| Attacker_Branch p Q ⇒
  ∃φ. strategy_formula
    (Attacker_Immediate p Q) (e - E 1 0 0 0 0 0 0 0) φ
    ∧ expressiveness_price φ ≤ e - E 1 0 0 0 0 0 0 0
using assms
proof (induction rule: attacker_wins.induct)
[...]
```

There are superficial differences due to the different medium. For instance, note that the inductive predicate in the Isabelle theory has a seventh case for `Attacker_Branch / [...]an` that does not exist in the “paper version” of Lemma 7.1. This is more natural for the “case ... of ...” formulation in the formalization, and addresses a technicality that Bisping & Jansen (2025) handle in the proof body.

The (mutual) induction on the formula structure to establish the distinctiveness of Strat_{∇} -formulas of Lemma 7.2 begins:¹²⁹

```

Lemma strategy_formulas_distinguish:
  <(strategy_formula g e φ →
    (case g of
      Attacker_Immediate p Q ⇒
        distinguishes_from φ p Q
    | Defender_Conj p Q ⇒
        distinguishes_from φ p Q
    | _ ⇒ True))
  ∧
  (strategy_formula_inner g e χ →
    (case g of
      Attacker_Delayed p Q ⇒
        (Q →S Q) → distinguishes_from (Internal χ) p Q
    | Defender_Conj p Q ⇒
        hml_srbb_inner.distinguishes_from χ p Q
    | Defender_Stable_Conj p Q ⇒
        (∀q. ¬ p → τ q)
        → hml_srbb_inner.distinguishes_from χ p Q
    | Defender_Branch p α p' Q Qa ⇒
```

¹²⁹  lemma `Strategy_Formulas.weak_spectroscopy_game.strategy_formulas_distinguish`

```

      (p  $\mapsto$  a  $\alpha$  p')
       $\longrightarrow$  hml_srb_inner.distinguishes_from  $\chi$  p (QUQa)
    | _  $\Rightarrow$  True))
  ^
  (strategy_formula_conjunct g e  $\psi$   $\longrightarrow$ 
    (case g of
      Attacker_Conjunct p q  $\Rightarrow$ 
        hml_srb_inner.distinguishes  $\psi$  p q
      | _  $\Rightarrow$  True)))
proof (induction rule:
  strategy_formula_strategy_formula_inner[...].induct
  [...])

```

For the other direction, the following expresses distinction completeness (Lemma 7.3) that attacks exist for formulas.¹³⁰

```

lemma distinction_implies_winning_budgets:
  assumes <distinguishes_from  $\varphi$  p Q>
  shows <attacker_wins (expressiveness_price  $\varphi$ )
        (Attacker_Immediate p Q)>


```


The main theorem Theorem 7.1 combines the previous facts and the upward-closedness of attacker winning budgets `win_a_upwards_closure`. This is quite straightforward. The following listing reproduces it in full to convey an idea of what Isar proofs look like.¹³¹

```

theorem spectroscopy_game_correctness:
  shows
    < $\exists \varphi$ . distinguishes_from  $\varphi$  p Q  $\wedge$  expressiveness_price  $\varphi \leq e$ >
     $\longleftrightarrow$  attacker_wins e (Attacker_Immediate p Q)>
proof
  assume
    < $\exists \varphi$ . distinguishes_from  $\varphi$  p Q  $\wedge$  expressiveness_price  $\varphi \leq e$ >
  then obtain  $\varphi$  where  $\varphi_{\text{spec}}$ :
    <distinguishes_from  $\varphi$  p Q> <expressiveness_price  $\varphi \leq e$ >
  by blast
  from distinction_implies_winning_budgets  $\varphi_{\text{spec}}$ (1) have
    <attacker_wins
      (expressiveness_price  $\varphi$ ) (Attacker_Immediate p Q)> .
  thus <attacker_wins e (Attacker_Immediate p Q)>
    using win_a_upwards_closure  $\varphi_{\text{spec}}$ (2) by simp
next
  assume <attacker_wins e (Attacker_Immediate p Q)>
  with winning_budget_implies_strategy_formula have
    < $\exists \varphi$ . strategy_formula (Attacker_Immediate p Q) e  $\varphi$ >

```

¹³⁰  lemma Distinction_Implies_Winning_Budgets.weak_spectroscopy_game.distinction_implies_winning_budgets

¹³¹  theorem Silent_Step_Spectroscopy.weak_spectroscopy_game.spectroscopy_game_correctness

```

    ∧ expressiveness_price  $\varphi \leq e$ 
  by force
hence
  < $\exists \varphi$ . strategy_formula (Attacker_Immediate p Q) e  $\varphi$ 
    ∧ expressiveness_price  $\varphi \leq e$ >
  by blast
thus < $\exists \varphi$ . distinguishes_from  $\varphi$  p Q ∧ expressiveness_price  $\varphi \leq e$ >
  using strategy_formulas_distinguish by fastforce
qed

```

In Chapter 2, we started our journey of connecting attacker strategies and distinguishing formulas in Theorem 2.3. In retrospect, it is just a corollary of this formalized theorem.

7.1.4 Complexity

The complexity of spectroscopy using the weak game is quite comparable to the strong spectroscopy (Theorem 5.2).

Theorem 7.2 (Weak spectroscopy complexity). *Given a transition system \mathcal{S} , the spectroscopy problem for the N^{weak} -spectrum can be solved by the game approach in exponential time and space with respect to the state space size $|\mathcal{P}|$.*

Proof. According to Theorem 7.1, we can solve the spectroscopy problem for the N^{weak} -spectrum by deciding the winning budgets of the weak spectroscopy game $\mathcal{G}_{\nabla}^{\mathcal{S}}$ on $\mathcal{S} = (\mathcal{P}, \text{Act}, \rightarrow)$. We instantiate the winning budget complexity of Lemma 4.5 for the case $d = 8$ with the size of \mathcal{G}_{∇} according to Definition 7.1.

The number of attacker positions $[\dots]_a$ (and their delayed $[\dots]_a^{\varepsilon}$ and branching $[\dots]_a^{\eta}$ variants) is bounded by $O(|\mathcal{P}| \cdot 2^{|\mathcal{P}|})$. The number of conjunction moves and defender conjunction positions $(\dots)_d$ is bounded by $|\mathcal{P}| \cdot 2^{|\mathcal{P}|}$, also for the stable variant $(\dots)_d^s$.

However, for the branching conjunction moves, we find a bound of $O(|\rightarrow| \cdot 2^{|\mathcal{P}|})$ per attacker delayed position (which is a slight over-approximation). Collectively, these moves reach $O(|\rightarrow| \cdot 3^{|\mathcal{P}|})$ defender branching positions $(p, \alpha, p', Q \setminus Q_{\alpha}, Q_{\alpha})_d^{\eta}$, due the three-coloring of states into $Q \setminus Q_{\alpha}$, Q_{α} and $\mathcal{P} \setminus Q$.

The maximal out-degree for attacker delayed positions of $O(|\rightarrow| \cdot 2^{|\mathcal{P}|})$ dominates that of other positions, in particular, of defender conjunction, stable conjunction, and branching positions with $O(|\mathcal{P}|)$ outgoing options.

This amounts to $o_{\rightarrow \nabla}$ in $O(|\rightarrow| \cdot 2^{|\mathcal{P}|})$ and to $|G_{\nabla}|$ in $O(|\rightarrow| \cdot 3^{|\mathcal{P}|})$. Inserting the parameters in the time bounds of Lemma 4.5 yields:

$$\begin{aligned}
 & O(\quad o_{\rightarrow \nabla} \quad \cdot \quad |G|^{2 \cdot d} \quad \cdot \quad (d^2 + |G|^{d-1} \cdot d) \quad) \\
 = & O(\quad (|\rightarrow| \cdot 2^{|\mathcal{P}|}) \quad \cdot \quad (|\rightarrow| \cdot 3^{|\mathcal{P}|})^{2 \cdot 8} \quad \cdot \quad (8^2 + (|\rightarrow| \cdot 3^{|\mathcal{P}|})^{8-1} \cdot 8) \quad) \\
 = & O(\quad |\rightarrow| \cdot 2^{|\mathcal{P}|} \quad \cdot \quad |\rightarrow|^{16} \cdot 3^{16|\mathcal{P}|} \quad \cdot \quad |\rightarrow|^7 \cdot 3^{7|\mathcal{P}|} \quad) \\
 \subseteq & O(\quad |\rightarrow|^{24} \quad \cdot \quad 3^{24|\mathcal{P}|} \quad).
 \end{aligned}$$

For space complexity, we arrive at $O(|\rightarrow|^8 \cdot 3^{8|P|})$. \square

The exponential out-degree is due to branching conjunction moves. That these would need exponentially many outgoing moves seems off: These moves are for η - and branching bisimilarity, which are known to be at the less expensive end of equivalence problems in the spectrum (sub-cubic by [Groote et al., 2017](#)). Frutos Escrig et al.’s (2017) branching bisimulation game is polynomially-sized. Thus, a derived reachability game of the weak spectroscopy game for branching bisimilarity in the spirit of Section 5.3.1 should, too, be polynomial in size if we apply clever optimizations. Section 7.3.1 will later show how to simplify the spectroscopy game to achieve this reduction of size around branching conjunctions.

7.2 Tackling Our Case Studies

We now return to our examples from Section 6.2, and see how the spectroscopy approach can handle the two, and thus support research in concurrency theory. For this project, we will use the [equiv.io](#) CCS dialect.

7.2.1 Parallelizing Compilers

In Example 6.4, we have discussed the example of *parallelizing compilation*. Bell (2013) reports on the search for a fitting notion of equivalence to prove the equivalence of sequential and parallelized program, P_{Seq} and P_{Para} .

We model the computation of A/B as a parallel process `Compute` that outputs one of the two options (`computeA/computeB`). In P_{Seq} , the computation has to happen before the visible output action `printOutput!` starts, followed by the printing of the computed value, `printA!` or `printB!`. In P_{Para} , the `printOutput!` happens in parallel, and the two processes synchronize on `join` before continuing to print the computed value. For both processes, the synchronizing actions, that is, `computeA`, `computeB`, and `join`, are restricted to internal use.

[Interactive model on equiv.io.](#)

```
Compute = computeA!Compute + computeB!Compute

P_Seq = (
  Compute
  | computeA.printOutput!printA!
  + computeB.printOutput!printB!
) \ {computeA, computeB}

P_Para = (
  Compute
  | printOutput!join!
  | computeA.join.printA!
```

```

    + computeB.join.printB!
  ) \ {computeA, computeB, join}

@compareSilent P_Para, P_Seq

```

The statement `@compareSilent P_Para, P_Seq` invokes the *weak spectroscopy* on the pair of processes.

As a cheapest distinguishing formula of P_{Para} from P_{Seq} , the spectroscopy reports $\langle \epsilon \rangle \langle \text{printOutput!} \rangle \langle \epsilon \rangle \wedge \{ \langle \epsilon \rangle \langle \text{printA!} \rangle \top, \langle \epsilon \rangle \langle \text{printB!} \rangle \top \}$. This formula disproves weak simulation plus weak readiness and everything above in the weak spectrum. The minimality means that all other notions must hold. These are the equivalences of contrasimilarity and stable bisimilarity, and below.

This nicely shows that our spectroscopy algorithm allows to survey the whole spectrum of possible equivalences between such small processes in milliseconds. Such a mechanized survey would certainly have facilitated the research behind Bell (2013).

7.2.2 τ -Abstraction and Failures

Section 6.2.2 has introduced a transition system with two states P_e and P_ℓ that perform a nondeterministic op-step. Only P_ℓ can correct the choice while idling. Though both processes are weak-failure-equivalent, their variants P_e^τ and P_ℓ^τ where idle-steps are relabelled to τ -steps are not.

To reconstruct this result on equiv.io, we first have to translate the processes of Figure 6.4 to CCS. We express P_e and P_ℓ as P_e and P_l through mutual recursion in the following model:

```

Ae = idle.Ae + a
Be = idle.Be + b
Al = idle.Bl + idle.Al + a
Bl = idle.Al + idle.Bl + b

P_e = op.Ae + op.Be
P_l = op.Al + op.Bl

```

[Interactive model on equiv.io.](http://equiv.io)

To implement the *hiding* of `idle` in CCS, we use a parallel process that provides unlimited `idle!`-actions to synchronize instead of the environment. Hence, P_e^τ and P_ℓ^τ are expressed as P_te and P_tl :

```

Idle = idle!Idle
P_te = (P_e | Idle) \ {idle}
P_tl = (P_l | Idle) \ {idle}

```

We can now use the derived equivalence checkers (along the lines of Section 5.3) to establish that P_e and P_l are weak-failure-equivalent, but P_te and P_tl are not:

```
@check weak-failure, P_e, P_l
> "States are equivalent."
@check weak-failure, P_te, P_tl
> "States are inversely preordered (only from right to left)."
```

How does this influence research in concurrency theory? Gazda et al. (2020, Corollary 9) claim that weak failure equivalence would be a congruence for the hiding operator. In light of our finding, this cannot be right.¹³²

We can go even further and decide *all* weak equivalences of our spectrum:

```
@compareSilent P_e, P_l
> "Equated by:
  weak-readiness
  stable-readiness"
@compareSilent P_te, P_tl
> "Equated by:
  stable-bisimulation"
```

This provides us with the general answer that notions can only be congruences for hiding if they are not below weak/stable readiness, or if they moreover fall into the hierarchy below stable bisimulation.

7.3 Variants

At this point in the thesis, it hopefully has become quite apparent how our approach can be varied with respect to game moves and HML hierarchies in order to add more equivalences or to cut resolution.

This section mainly presents two tricks that are used in the implementation of [equiv.io](#): Section 7.3.1 shows how to get rid of the subset construction in branching conjunctions. Section 7.3.2 presents a modification to add stable revivals, stable failure traces and stable ready traces as notions. In Section 7.3.3, we close with some hints how even more nuances in weak equivalences could be covered.

7.3.1 Optimizing Branching Conjunctions

Bisping & Jansen (2025) show how to reduce the out-degree o_{\rightarrow} of the weak spectroscopy game to be linear. For this, we reformulate the branching conjunction part of the game to be closer to the operational Definition 6.7 of branching bisimilarity. We can still solve the main spectroscopy problem, but lose some resolution about the number of nested conjunctions.

If we read Definition 6.7 directly as a game, it differs from the branching conjunction moves in Definition 7.1, because the latter require the attacker to name as Q_α ex-ante which q' to challenge directly and which ones only after

¹³² The bug in the otherwise intriguing paper (Gazda et al., 2020) is that the modal logic of weak failures is not closed under adding of weak observation sequences at $\langle \varepsilon \rangle$ -operators: $\langle \varepsilon \rangle \langle a \rangle \langle \varepsilon \rangle \wedge \{ \neg \langle \varepsilon \rangle \langle a \rangle \} \in \mathcal{O}_{WF}$, but $\langle \varepsilon \rangle \langle a \rangle \langle \varepsilon \rangle \wedge \{ \neg \langle \varepsilon \rangle \langle a \rangle \langle \varepsilon \rangle \langle a \rangle \}$ is too strong for weak failure equivalence. Their requirement (ABS) together with the definition of \mathcal{T}_H^{-1} would prescribe that the greatest logic to characterize weak failures contains both, for hiding to be a congruence. Their Corollary 9 assumes in error that this kind of pumping were sound for weak failures.

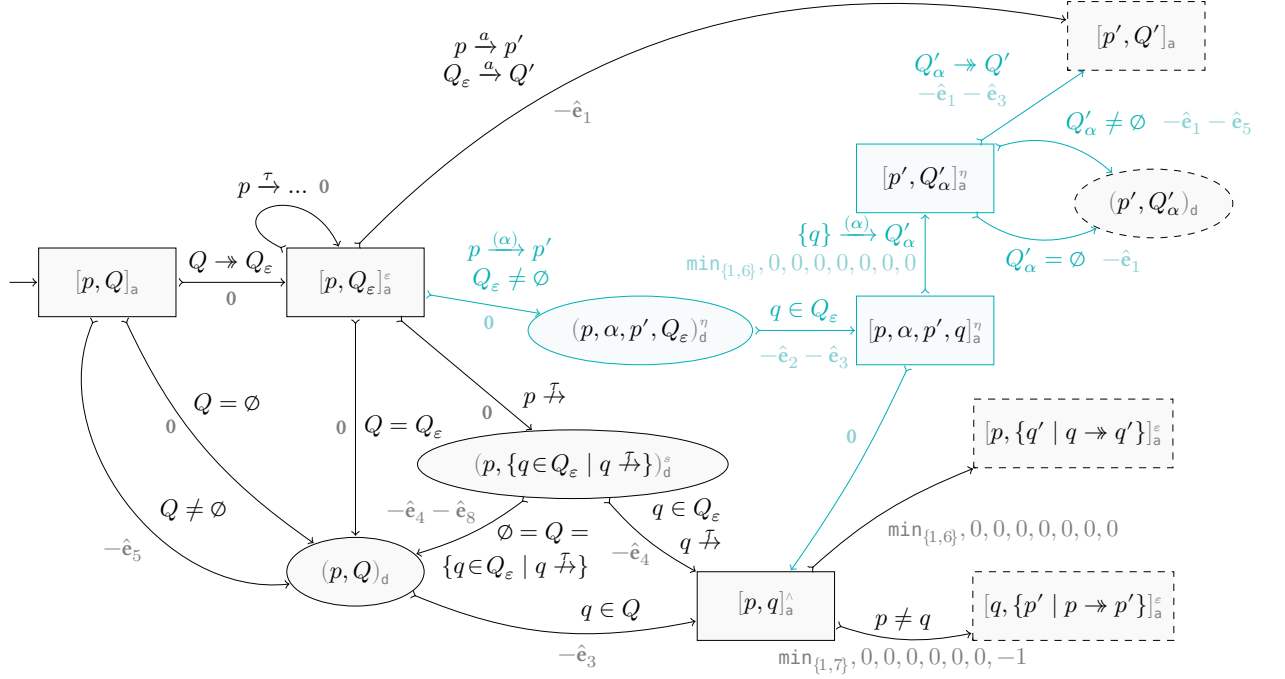


Figure 7.3: Schematic *simplified* weak spectroscopy game $\mathcal{G}_{\hat{\nabla}}$ with adapted branching conjunction section (the teal part).

the α step, and to have one continuation for the whole Q_α group. The *simplified* weak spectroscopy game $\mathcal{G}_{\hat{\nabla}}$ in Figure 7.3 rephrases this part to match the operational characterization. (This might not be “simpler” conceptionally, but it will be for the algorithm.)

Intuitively, the simplified game part encodes nested conjunctions of the form $\bigwedge\{(\alpha) \wedge \Psi', \psi_1, \dots\}$ or the cheaper form $\bigwedge\{(\alpha) \langle \varepsilon \rangle \wedge \Psi', \psi_1, \dots\}$. The Ψ' are the formulas from after branching observation moves $[p, \alpha, p', q]_a^\eta \twoheadrightarrow [p', Q'_\alpha]_a^\eta \twoheadrightarrow [p', Q']_a$, while the ψ_i come from the resets $[p, \alpha, p', q]_a^\eta \twoheadrightarrow [p, q]_a^\wedge$. In Bisping & Jansen (2025), we discuss this game’s strategy formulas in detail and prove its correctness, resulting in the following theorem.¹³³

¹³³ Credit for the proof details goes to Jansen.

Theorem 7.3 ($\hat{\mathbb{N}}^{\text{weak}}$ characterization). *Let the simplified weak notions $\hat{\mathbb{N}}^{\text{weak}}$ be the union of $\mathbb{N} \times \{0\} \times \mathbb{N}^6$ and $\mathbb{N} \times \{\infty\} \times \{0, \infty\} \times \mathbb{N} \times \{0, \infty\} \times \mathbb{N}^3$. Let simplified expressiveness prices be defined by rounding up the prices of Definition 6.11: $\widehat{\text{expr}}^{\text{weak}} = \min\{N \in \hat{\mathbb{N}}^{\text{weak}} \mid \text{expr}^{\text{weak}}(\varphi) \leq N\}$. Then, on the simplified weak game $\mathcal{G}_{\hat{\nabla}}$ of Figure 7.3:*

For all $N \in \hat{\mathbb{N}}^{\text{weak}}$, $p \in \mathcal{P}$, $Q \in 2^{\mathcal{P}}$, the following are equivalent:

- *There exists a formula $\varphi \in \text{HML}_{\text{SRBB}}$ with price $\widehat{\text{expr}}^{\text{weak}}(\varphi) \leq N$ that distinguishes p from Q .*
- *Attacker wins $\mathcal{G}_{\hat{\nabla}}^S$ from $[p, Q]_a$ with energy N .*

Complexity-wise, for the simplified game, $\mathcal{G}_{\widehat{\nabla}}$, we have just $|G_{\widehat{\nabla}}| \in O(|\rightarrow| \cdot 2^{|\mathcal{P}|})$ and also $o_{\widehat{\nabla}} \in O(|\rightarrow|)$. Following the same argument as in Theorem 7.2, deciding the whole game still has exponential time complexity of $O(|\rightarrow| \cdot (|\rightarrow| \cdot 2^{|\mathcal{P}|})^{16} \cdot (|\rightarrow| \cdot 2^{|\mathcal{P}|})^7) = O(|\rightarrow|^{24} \cdot 2^{23|\mathcal{P}|})$, and space complexity $O(|\rightarrow|^8 \cdot 2^{8|\mathcal{P}|})$. But these are much lower bounds than in the original game \mathcal{G}_{∇} , where we found $O(|\rightarrow|^{24} \cdot 3^{24|\mathcal{P}|})$ for time and $O(|\rightarrow|^8 \cdot 3^{8|\mathcal{P}|})$ for space.

The practical difference is huge on transition systems exposing relevant branching-degree with respect to internal behavior such as the initial Peterson system of Figure 1.2: With the optimization, the tool solves it in fractions of a second, as reported in Table 8.4. Without it, the exponential conjunctions lead to a game with 121,773 moves, taking 90 seconds.

Moreover, we can again use the trick to work with *flattened energies*, according to Lemma 4.6. After all, $\mathcal{G}_{\widehat{\nabla}}$ itself is only correct with respect to a simplified spectrum according to Theorem 7.3. If we bound the energy lattice to $\{0, 1, \infty\}^8$ the size of Pareto fronts is decoupled from the game size. This further improves space complexity to $O(|\rightarrow| \cdot 2^{|\mathcal{P}|})$ and overall time complexity to $O(|\rightarrow| \cdot (|\rightarrow| \cdot 2^{|\mathcal{P}|})^{16}) = O(|\rightarrow|^{17} \cdot 2^{16|\mathcal{P}|})$.

7.3.2 Covering Revivals and Decorated Traces

In the weak spectrum of Section 6.3, we left out the notions of revivals, failure traces, and ready traces, which we had included in the strong spectrum of Section 3.2.2. Their stable variants are relevant to the CSP community (see Roscoe, 2009).

As discussed in Example 3.5, these notions need to differentiate between a deepest “revival” conjunct and other positive conjuncts. Thus, these equivalences need an additional dimension for expr-measurements, and an even more sophisticated handling of conjunctions in the game.

Figure 7.4 illustrates how one could incorporate revivals into stable conjunctions, analogously to Chapter 5. Note, that we now have two kinds of conjunct positions: for the stable non-revival context and for the other contexts. The maximal depth of conjuncts is still managed by dimension 6. But stable non-revival conjuncts receive a new dedicated dimension 7 to bound their depth. The previous dimensions 7 and 8 now come 8th and 9th.

In this game, stable revivals end up at $(\infty, 0, 0, 1, 0, 1, 0, 1, 1)$, stable failure traces at $(\infty, 0, 0, 1, 0, \infty, 0, 1, 1)$, and stable ready traces at $(\infty, 0, 0, 1, 0, \infty, 1, 1, 1)$, analogously to their strong counterparts in Figure 3.8. Of course, one could again use the “look-ahead trick” of Section 5.2.2 to reduce the number of partitions to consider.

In fact, the implementation on equiv.io does employ this formulation, thereby actually using a 9-dimensional game with richer stable conjunctions. Although this thesis remains short on providing a theorem for the game that contains the stable revival move, tests in the tool (and the theorems for the strong spectrum) suggest correctness of the variant in Figure 7.4.

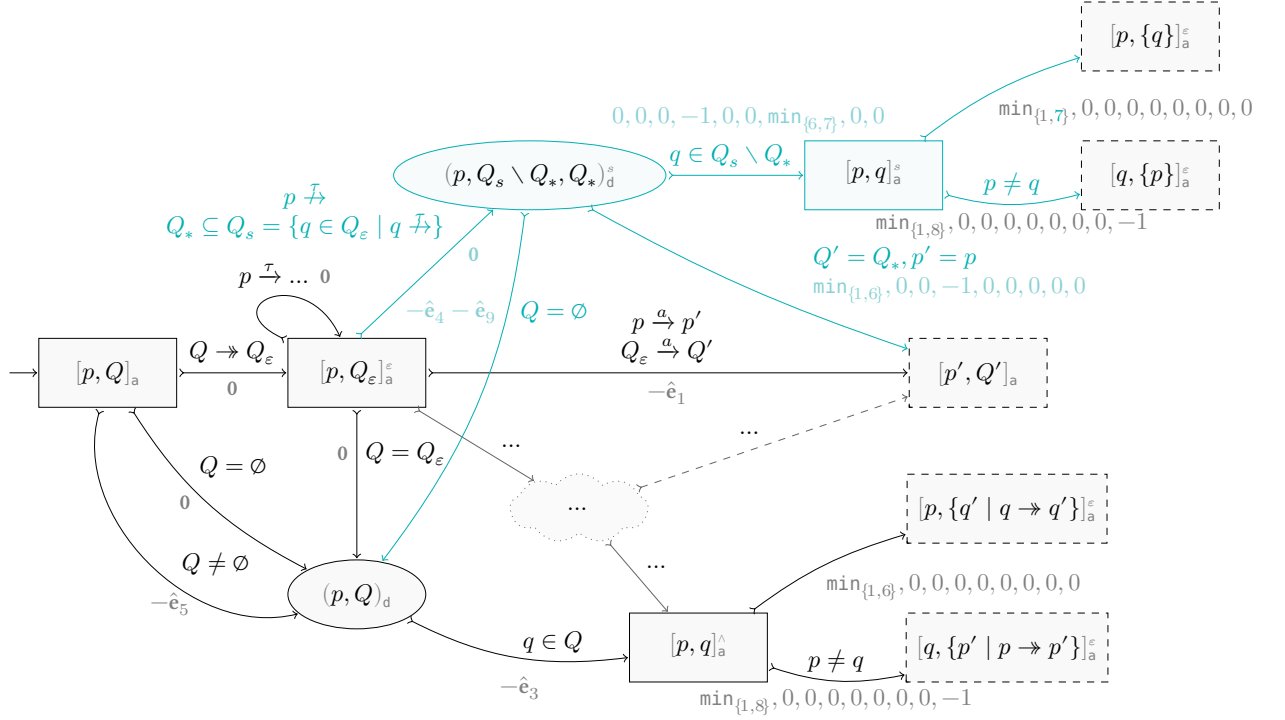


Figure 7.4: Modifying stable conjunction moves to include revivals (teal part). The branching bisimulation part is left out.

7.3.3 Extending to Other Equivalences

At this point, we have covered most interesting parts of strong and weak spectrum. But there are still more notions one could reach for.

Divergence and completed observations. Logic and game, as we have presented them, are blind to *divergence* and *completed observations*. Van Glabbeek (1993) uses additional modalities: Δ for divergence with $\llbracket \Delta \rrbracket := \{p \mid p \xrightarrow{\omega}\}$; 0 for completed observations with $\llbracket 0 \rrbracket := \{p \mid \forall \alpha \in \text{Act}. \xrightarrow{\alpha}\}$ (and $\lambda := 0 \vee \Delta$).

Bisping & Jansen (2024) decide against including these modalities in the game. At least on finite-state systems, they may be understood to be *special action observations*. Divergence and completion can be added through pre-processing into a system \mathcal{S} before turning to our game of equivalence questions on \mathcal{S}' .

For 0, the transformation from \mathcal{S} to \mathcal{S}' is obvious: Add a $p \xrightarrow{\checkmark} \perp$ to the transition system for each $p \in \mathcal{P}$ where $p \not\xrightarrow{a}$ for every $a \in \text{Act}_{\bullet}$ (with \checkmark and \perp fresh). Then $\llbracket \langle \checkmark \rangle \top \rrbracket^{\mathcal{S}'} = \llbracket 0 \rrbracket^{\mathcal{S}}$.

For divergence on finite-state systems, one may use an argument from Groote et al. (2017): Add a state \perp , an action $\delta \notin \text{Act}$ and transitions $p \xrightarrow{\delta} \perp$ to the transition system for each $p \in \mathcal{P}$ that lives on a τ -cycle $p \xrightarrow{\tau^+} p$. Then $\llbracket \langle \varepsilon \rangle \langle \delta \rangle \top \rrbracket^{\mathcal{S}'} = \llbracket \Delta \rrbracket^{\mathcal{S}}$. Fokkink et al. (2019) define a unary *diverges-*

while operator $\Delta\varphi$ to characterize divergence-preserving branching bisimilarity; this operator is then naturally expressed as branching conjunction $\langle\varepsilon\rangle \wedge \{(\delta)\top, \varphi\}$.

For infinite systems, divergence is more tricky. Just like infinite traces, it depends on the possibility of characterizing infinite-duration attacks.¹³⁴ On the game level, for infinite plays to be winnable by the attacker, the game must have a parity-game winning condition: The attacker wins a subgame about divergence-distinction if they can make infinitely much τ -progress, but the defender cannot. Such a richer game model is the route taken by Frutos Escrig et al. (2017) to characterize various divergence-aware bisimilarities.

Behavioral congruences. Famously, many of the unstable weak equivalences need to be refined in order to be congruences for CCS with choice “+” (cf. Gazda et al., 2020; Sangiorgi, 2012). For instance, *rooted* weak bisimulation congruence is achieved by allowing an $\langle\varepsilon\rangle\langle\tau\rangle\dots$ -observation at the outermost level of the HML characterization. Otherwise, weak bisimulation formulas have no $\langle\tau\rangle$ -parts (see Figure 6.7).

This thesis is not going deeper into congruences. But let us note that, clearly, the weak spectroscopy game could be extended by some prefix to enable such special observations for the attacker at the beginning if rootedness is desired.

Coupled simulation. As mentioned in Section 6.4, we have glimpsed over coupled similarity. It can be thought of as the syntactic combination of weak similarity and contrasimilarity in the sense that each weak conjunction must either contain purely positive or purely negative conjuncts, as outlined in Bisping & Montanari (2024). This possibility does not align nicely with our game formulation because it means that the attacker has to dedicate a conjunction to positivity or negativity before entering, and not make up their mind on the fly whether $q \in Q$ are addressed positively or negatively. If desired, this could be treated by a new dimension in modal conjunctions and game positions. Who urgently needs to compute coupled similarity relations, may otherwise fall back to the coupled simulation game by Bisping & Nestmann (2019).

7.4 Discussion

This chapter has shown that the spectroscopy approach can readily be extended to handle weak behavioral equivalences as well.

The weak spectrum, gamified. With the weak spectroscopy game, Bisping & Jansen (2024) provide the *first generalized game characterization of the silent-step spectrum*.

Previously, there have only been partial characterizations of individual equivalences: Frutos Escrig et al. (2017) treat the diamond of weak, delay,

¹³⁴ And not just unbounded attack strategies as we support them now.

η and branching bisimilarity. Bisping et al. (2020) and Bisping & Montanari (2021, 2024) cover the area of coupled notions and contrasimilarity. Tan (2002b) describes subset-construction variants of the bisimulation game for weak trace equivalence and stable failure equivalence. With the weak spectroscopy game, we have moved beyond such individual equivalence games in order to achieve genericity, covering both linear-time and branching-time as well as stable and unstable notions.

In principle, this chapter only executes Idea 3 to translate modal constructions to game moves appearing in HML_{SRBB} , which follows Idea 10. The important trick is to weaken the attacker according to Idea 11, but this departs from prior approaches in weakened games and involves many technicalities. Therefore, it is relieving to have the Isabelle formalization of Section 7.1.3 by Barthel et al. (2025) to verify the construction.

Analyzing systems. The mini case studies of Section 7.2 show how the spectroscopy automates away the tedious kind of work of finding out the precise relationship between transformed processes that went, for instance, into Bell (2013). Similar searches for weak bisimilarities have happened around Parrow & Sjödin (1992) and Nestmann & Pierce (2000), where coupled similarity turned out to be the most fitting for encodings between models of differing synchronicity.¹³⁵

Concise distinguishing formulas can be interesting diagnostic information to compare models. Like Martens & Groote (2024), we find minimal-depth distinguishing formulas for branching bisimilarity without the need for a special *until* operator, but we solve the problem for *all* weak notions at once. Horne et al. (2023) report how inequivalence with respect to more distinctive modal logics can reveal privacy vulnerabilities in ePassports, which have been overlooked in a purely trace-based view. For a full understanding of a system's relationship to its specification, it is often helpful to pinpoint exactly how difficult it would be for an attacker to tell the two apart.

To be continued. Some applications would demand more specialized modal logics. As outlined in Section 7.3.3, matching adaptations in game and semantic model are usually straightforward thanks to the clear connection of productions in the modal grammar and mechanics in the spectroscopy game.

But we have assembled enough theory to perform generalized equivalence checking on common models of concurrent systems. Time for Part IV to move beyond theory! In the next chapter, we will return to the initial Example 1.1 of Peterson's mutex and explain how the prototype implementation of the weak spectroscopy algorithm handles it.

¹³⁵ Nice accounts of Nestmann's search can be found in Barwell et al. (2022) and in the historical remarks of Bisping et al. (2020).

Part IV

... and Beyond

8 Implementations

All the nice theory of preceding chapters also works *in practice*. This chapter revisits the core parts of the thesis by discussing how they tie together in a tool implementation.

The tool, [equiv.io](#), will be presented in Section 8.1. We demonstrate its use through the Peterson’s mutex example. Section 8.2 shows that the game approach lends itself to many things: To explain equivalence notions in a computer game, to extend existing tools, and to parallelize equivalence checking through GPUs. In Section 8.3, we compare our implementations to similar tools in the lineage of the “Concurrency Workbench.”

Related publications. Conference artifacts of [equiv.io](https://artifacts.cavi.ac.uk/equiv.io) (reviewed by the artifact evaluation committees of TACAS'21 and CAV'23) are archived as *Linear-time-branching-time spectroscopy* (Bisping, 2021, 2023a). Some paragraphs of discussion in this chapter have already appeared in Bisping et al. (2022) and Bisping (2023b).

8.1 Prototype: *equiv.io*

The “Linear-time-branching-time spectroscoper” at equiv.io is a small web tool to check equivalence and preorder relations on CCS processes. Figure 8.1 shows a screenshot. In this section, we will discuss its usage (Section 8.1.1), apply it to the Peterson example (Section 8.1.2), examine the tool structure (Section 8.1.3), and benchmark its backend (Section 8.1.4).

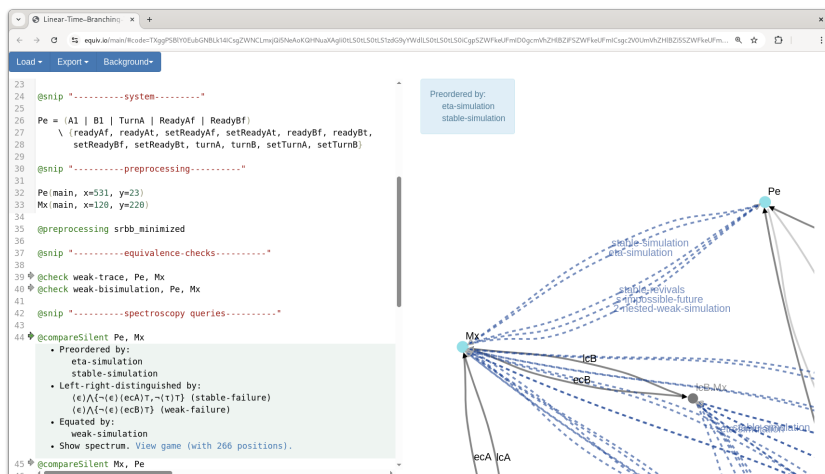


Figure 8.1: Screenshot of equiv.io.

8.1.1 Usage

The standard workflow of equiv.io is to specify processes in CCS and then compare them with respect to equivalence spectra. This is mainly achieved by writing text into the source editor on the left. The specific order of definitions, queries, and declarations in the source generally makes no difference.

Process syntax. CCS processes are written according to the grammar in Definition 2.2, with the concrete syntax for subterms following Table 8.1.

Table 8.1: Concrete ASCII syntax for CCS terms

Construct	Tool syntax	CCS
Input prefix	$a.P$	$a.P$
Output prefix	$a!P$	$\bar{a}.P$
Internal action	$\text{tau}.P$	$\tau.P$
Null process	\emptyset	0
Recursion	P_{Name}	P_{Name}
Choice	$P_1 + P_2$	$P_1 + P_2$
Parallel	$P_1 \mid P_2$	$P_1 \mid P_2$
Restriction	$P \setminus \{a_1, a_2\}$	$P \setminus \{a_1, a_2\}$

Literals for actions and process names combine Latin letters, numbers, and underscores in the usual way.

Top-level process definitions are written $X = P$, expressing that $\mathcal{V}(X) := P$ in the semantics (Definition 2.3). The right-hand pane shows the resulting transition system and output (cf. Figure 8.1).

The syntax tree can be clarified by parentheses “(. . .)”. Otherwise, the parser reads prefix “ $a._$ ” with highest operator precedence, then restriction “ $_ \setminus \{_\}$ ”, then choice “ $+$ ” and parallel “ \mid ” at equal level. In case of ambiguity, it assumes parenthesization from the right.

Equivalence queries. Queries for the behavioral equivalences between states are formulated in the source editor as well and are started by clicking on the arrows that pop up in the gutter. Output will appear right below the query in the editor. The standard queries are:

- @compare P_1, P_2 – Perform a *spectroscopy* on P_1 and P_2 with respect to the *strong spectrum* using the game of Chapter 5. The output will have four items, relative to the strong spectrum.
 1. The *strongest preorders* to relate P_1 to P_2 ;
 2. *Cheapest formulas to distinguish* P_1 from P_2 (and the smallest observation language they are part of);
 3. The list of *finest equivalences* to equate P_1 and P_2 .
 4. A *visualization* of the result on the whole spectrum. Moreover, there will be a link to a <https://edotor.net/-visualization> of the

game graph used to obtain the result. The naming of notions in the output follows Table 8.2.

- @compareSilent P1, P2 – Perform a *spectroscopy* on P1 and P2 with respect to the *weak spectrum*, treating silent steps along the lines of Chapter 7. The output is analogous to @compare.
- @check equivalence-name, P1, P2 – Checks for (mutual) preordering between P1 and P2 with respect to individual notion equivalence-name. The name of the notion to be checked must be one of the ones in Table 8.2.

Interacting with the output. Each output item can be clicked. The transition system view then displays the preorders, equivalences, and distinctions that have been found between states.

States in the transition system may be dragged to change their position. The layout information is persisted in the source, as described in the next list.

Layout and preprocessing. The source can also contain layout information and prescribe some preprocessing for processes.

- P(main) – Highlight process P in the transition system and prune other sub-processes (unless they are reached from P). Multiple processes may be declared to be main.

Table 8.2: Names of supported notions in *equiv.io*.

Strong variant	Weak variants	Stable / stability-respecting variants
enabledness	weak-enabledness	
trace	weak-trace	
failure	weak-failure	stable-failure
revivals		stable-revivals
readiness	weak-readiness	stable-readiness
failure-trace		stable-failure-trace
ready-trace		stable-ready-trace
impossible-future	weak-impossible-future	s-impossible-future
possible-future	weak-possible-future	
simulation	weak-simulation	stable-simulation
ready-simulation	weak-ready-simulation	s-ready-simulation
2-nested-simulation	2-nested-weak-simulation	
bisimulation	contrasimulation	stable-bisimulation
	weak-bisimulation	sr-delay-bisimulation
	delay-bisimulation	sr-branching-bisimulation
	eta-bisimulation	
	branching-bisimulation	

- `P(x=100, y=100)` – Annotate process `P` to be displayed at certain coordinates.
- `"0" (x=100, y=200)` – The annotations may also refer to subprocesses. The CCS expressions are wrapped in `"..."`. They must be verbatim the string representation the tool uses for the normalized process.
- `@preprocessing method1, method2...` – Apply preprocessing to the transition system after translation of the CCS term (that is, before presentation and queries happen). This will affect the whole transition system, but tries to preserve processes that have been marked as `main`. The order of processing steps can make a difference. The supported methods are:
 - `weakness_saturated` – Replace the transition relation with *weak transitions*. In effect, there will be a transition whenever the original system allows $\rightarrow^{(\alpha)} \rightarrow$.
 - `tauloop_compressed_marked` – Collapse states on τ -loops and mark them with a δ . (This follows the thought of how to make equivalence queries divergence-respecting from Section 7.3.3.)
 - `bisim_minimized` – Merge states that are strongly bisimilar.
 - `srbb_minimized` – Merge states that are stability-respecting branching-bisimilar (enforces tau-loops precisely on divergent states).
- `@comment "My comment"` – Any `@something`-tag without features can serve to add comments in the model.

8.1.2 Application to Peterson’s Mutual Exclusion Protocol

Let us go through the whole process of using the tool once to tackle the example of Peterson’s mutual exclusion protocol, which has already been previewed in Example 1.1. Thereby, we settle the question with respect to which equivalences the protocol correctly implements the specification of mutual exclusion. We follow the presentation of how to model this protocol in CCS from Aceto et al. (2007, Chapter 7), with action names chosen to align with van Glabbeek (2023).

We *specify* mutual exclusion as a system `Mx` of two alternating users `A` and `B` entering their critical section `ecA` / `ecB` and leaving `lcA` / `lcB` before the other may enter. Aceto et al. (2007, Equation 7.1) suggest the following specification in CCS:

Interactive model on equiv.io. `Mx = ecA.lcA.Mx + ecB.lcB.Mx`

Can one come up with a process where two subprocesses run a protocol such that the overall system is somewhat equivalent to this specification? Peterson (1981) proposes a protocol that can be summarized by the following pseudocode:

```

ready = {"A": false, "B": false}
turn = "A"

def process(ownId, otherId):
    while true:                                # PC = 1
        ready[ownId] = true
        turn = otherId

        do # wait...                            # PC = 2
            until (ready[otherId] == false || turn == ownId)

        print "enter critical #{ownId}"        # PC = 3
        # critical section goes here.
        print "leave critical #{ownId}"
        ready[ownId] = false

process("A", otherId = "B") || process("B", otherId = "A")

```

The two processes share three variables: `ready["A"]` expresses whether process A is ready to enter its critical section; `ready["B"]` the same for B. In turn, *both* processes try to write whose turn it is to enter, A or B. Assuming sufficiently consistent memory, the protocol works because each process will only enter the critical section if no other process is waiting or if it has been yielded the turn (by the other process). The critical scenario of both processes entering *symmetrically* is resolved because the race condition on the turn-write will flip a coin in such situations.

To express the shared-memory protocol in the message-passing paradigm of CCS, we must model the storage as processes that run in parallel with the main model. We do this as Aceto et al. (2007): In the following the process `ReadyAf` corresponds to `ready["A"] = false`, and `ReadyAt` to `ready["A"] = true`. The current state can be either read (`readyAf / readyAt`) or written (`setReadyAf / setReadyAt`).

```

ReadyAf = readyAf!ReadyAf + setReadyAf.ReadyAf + setReadyAt.ReadyAt
ReadyAt = readyAt!ReadyAt + setReadyAf.ReadyAf + setReadyAt.ReadyAt

ReadyBf = readyBf!ReadyBf + setReadyBf.ReadyBf + setReadyBt.ReadyBt
ReadyBt = readyBt!ReadyBt + setReadyBf.ReadyBf + setReadyBt.ReadyBt

TurnA = turnA!TurnA + setTurnA.TurnA + setTurnB.TurnB
TurnB = turnB!TurnB + setTurnA.TurnA + setTurnB.TurnB

```

Each main process iterates through the phases 1, 2, and 3, corresponding to `PC = 1, 2, 3` in above pseudocode. In 1, they set their ready and yield the turn; in 2, they wait until they hear that the other's ready is false or that it is their

turn; in 3, they enter and leave their critical section, unset their ready, and return to phase 1.

```
A1 = setReadyAt!setTurnB!A2
A2 = readyBf.A3 + turnA.A3
A3 = ecA.lcA.setReadyAf!A1

B1 = setReadyBt!setTurnA!B2
B2 = readyAf.B3 + turnB.B3
B3 = ecB.lcB.setReadyBf!B1
```

Peterson's protocol is the parallel composition of A1, B1 and the storage, restricting communication with the memory:

```
Pe = (A1 | B1 | TurnA | ReadyAf | ReadyBf)
    \ {readyAf, readyAt, setReadyAf, setReadyAt, readyBf, readyBt,
       setReadyBf, setReadyBt, turnA, turnB, setTurnA, setTurnB}
```

If you enter above listings, you will notice that the transition system view is cluttered by cycles for the subprocesses. These can be removed from the system output by declaring Pe and Mx as the main processes:

```
Pe(main, x=900, y=340)
Mx(main, x=120, y=220)
```

To further clarify the transition graph, we can minimize it with respect to stability-respecting branching bisimilarity, resulting in the transition system of Figure 1.2:

```
@preprocessing srbb_minimized
```

We can now verify that Pe and Mx allow for the same weak traces, which, for instance, rules out bugs where both enter, ecA and ecB, after each other without the other leaving.

```
@check weak-trace, Pe, Mx
> "States are equivalent."
```

But the processes are *not* weakly bisimilar, as tested by:

```
@check weak-bisimulation, Pe, Mx
> "States are not preordered (nor equivalent)"
```

To get the full picture, we run a silent-step spectroscopy:

```
@compareSilent Pe, Mx
```

This yields output (also to be seen in Figure 8.1):

- Preordered by:
 - eta-simulation
 - stable-simulation
- Left-right-distinguished by:
 - $\langle \epsilon \rangle \wedge \{ \neg \langle \epsilon \rangle \langle \text{ecA} \rangle \top, \neg \langle \tau \rangle \top \}$ (stable-failure)
 - $\langle \epsilon \rangle \wedge \{ \neg \langle \epsilon \rangle \langle \text{ecB} \rangle \top \}$ (weak-failure)
- Equated by:
 - weak-simulation

The maintained notions can also be marked in an overlay on the weak spectrum, as shown in Figure 1.3.

The weak failure $\langle \epsilon \rangle \wedge \{ \neg \langle \epsilon \rangle \langle \text{ecB} \rangle \top \}$ can be understood to point out that *Pe exceeds Mx* in that it can reach a state without visible activity where *ecB* is impossible.

... thinking about it, this behavior might be okay for a mutual exclusion protocol, might it not? Why should the outside observer need to be notified about the participating processes making up their minds? So, maybe, our specification *Mx* is too strict. Let us try another specification of mutual exclusion where we leave it as an *iterated internal choice* of the system which participant may enter the critical section:

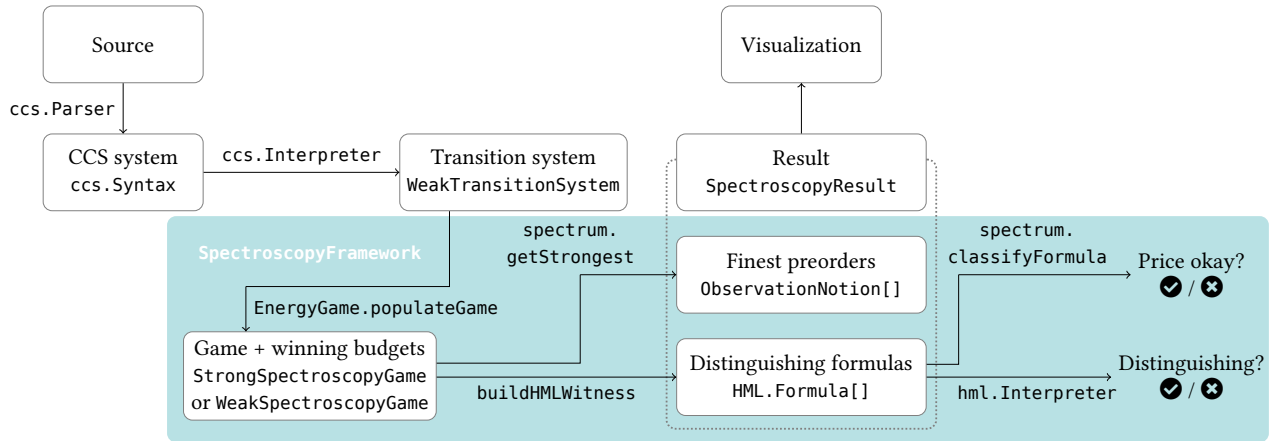
```
@comment "Internal-choice mutex"

MxIC(main, x=120, y=0)
MxIC = tau.ecA.lcA.MxIC + tau.ecB.lcB.MxIC

@compareSilent Pe, MxIC
```

Pe aligns much better to the *MxIC*-specification! “@compareSilent *Pe*, *MxIC*” returns:

- Preordered by:
 - eta-simulation
 - weak-ready-simulation
 - s-ready-simulation
- Left-right-distinguished by:
 - $\langle \epsilon \rangle \langle \text{ecA} \rangle \langle \epsilon \rangle \wedge \{ \neg \langle \epsilon \rangle \langle \text{lcA} \rangle \langle \epsilon \rangle \langle \text{ecA} \rangle \top, \neg \langle \tau \rangle \top \}$ (s-impossible-future)
 - $\langle \epsilon \rangle \langle \text{ecB} \rangle \langle \epsilon \rangle \langle \text{lcB} \rangle \wedge \{ \neg \langle \epsilon \rangle \langle \text{ecB} \rangle \top \}$ (delay-bisimulation)
 - $\langle \epsilon \rangle \langle \text{ecA} \rangle \langle \epsilon \rangle \wedge \{ \neg \langle \epsilon \rangle \langle \text{lcA} \rangle \langle \epsilon \rangle \langle \text{ecA} \rangle \top \}$ (weak-impossible-future)
- Equated by:
 - stable-readiness

Figure 8.3: Data transformation flow in *equiv.io*

however, we cannot adequately treat more general fairness considerations for Peterson’s protocol, as van Glabbeek (2023) explicates.

This is what the spectroscopy has taught us: Peterson’s mutual exclusion protocol is *more similar* to repeated internal choice (MxIC) than to the specification Mx from Aceto et al. (2007). However, the two are not bisimilar in any sense, since, in Pe, local progress from one iteration may influence the next.

8.1.3 Program Structure

The core of *equiv.io* aligns quite closely to the spectroscopy framework outlined in Figure 5.1. In this subsection, we take a quick look at how implementation and definitions in this thesis correlate.

Figure 8.3 shows core transformations that happen in the process of analyzing the equivalences for a pair of processes:

1. **Parsing.** `ccs.Parser` transforms source into an abstract syntax tree object `ccs.Syntax.Definition`, along the lines of Definition 2.2 with the syntax of Section 8.1.1.
2. **Interpretation.** `ccs.Interpreter` applies the operational semantics of Definition 2.3¹³⁶ to construct a `ts.WeakTransitionSystem`, which supports silent-step transitions of Section 6.1.1. Also the preprocessing of Section 8.1.1 is applied, the soundness of which follows Lemma 5.4 (and its analogues for the weak spectrum).
3. **Spectroscopy.** The trait `spectroscopy.SpectroscopyFramework` orchestrates the spectroscopy pipeline. Its abstract parts are instantiated by `spectroscopy.StrongSpectroscopy` and `spectroscopy.WeakSpectroscopy` for the respective spectroscopy variants of Chapter 5 and Chapter 7. In particular, they facilitate the following steps in `SpectroscopyFramework.decideAll`:

¹³⁶ There is a minor semantical difference: The *equiv.io* interpreter flattens process restriction in recursion. This leads to processes like $P = a. P \setminus \{b\}$ having a finite process graph instead of an infinite one (cf. Aceto et al., 2007, Exercise 2.9).

1. The spectroscopy defines a *spectroscopy game* (e.g. `spectroscopy.StrongSpectroscopyGame`). The game must be a `game.EnergyGame`, inheriting a decision procedure along the lines of Algorithm 4.1 from trait `game.GameLazyDecision`. This computation is invoked together with the game graph construction through `populateGame`.
2. The spectroscopy provides an `hml.Spectrum` object which is used to interpret the game result in a hierarchy of `hml.ObservationNotions` to pick the strongest preorders to relate compared processes. The specifics of Chapter 3 and Chapter 6 are implemented by `hml.StrongObservationNotion` and `hml.WeakObservationNotion`.
3. The spectroscopy implements a `buildHMLWitness`-method, constructing *strategy formulas* from attacker-won budgets of Strat_Δ in Definition 5.2 and Strat_∇ in Definition 7.2.
4. **Validation (optional).** If the user demands, the procedure `SpectroscopyFramework.decideAll` continues to construct cheapest distinguishing formulas for the query with `buildHMLWitness`.
 - `hml.Interpreter` checks that each formula is indeed true for one process and false for the other, applying the semantic HML game of Section 2.4.2. If the formula is not distinguishing, an exception is thrown.
 - `classifyFormula` on the specific `Spectrum` object determines the expressiveness prices of formulas (implemented as in Remark 3.1 and Definition 6.11 by the `formulaObsNotion` functions in `hml.StrongObservationNotion` / `hml.WeakObservationNotion`). If the formula price exceeds the budget predicted by the game or is unexpectedly cheap, this constitutes an error.

Error reports would point to incorrectness of the specific spectroscopy, not to user mistakes. Thus, the validation does not affect the core decision procedure, but increases confidence that a specific output is sound.

If no formula construction is requested, the implementation in `SpectroscopyFramework.decideAll` works on *flattened energies* (Definition 4.13), generally leading to better performance.

5. **Presentation.** `SpectroscopyFramework.decideAll` collects the finest preorders and coarsest distinctions, optionally together with their witness formulas in a `Spectroscopy.Result` object from `spectroscopy.Spectroscopy`. This object helps front-end layers of the tool interpret the output as spectra, equivalences, and distinguishing formulas.

Another trait `spectroscopy.EquivalenceChecking` follows the second path of Figure 5.1 to decide individual equivalences through reachability

games. We will not go into detail here. The core feature is to derive a game `game.MaterializedEnergyGame` as in Definition 4.4. `game.WinningRegionComputation` determines its winner according to Algorithm 2.1.

Besides the core-flow, there are some small additional diagnostic features. For instance, `spectroscopy.SpectroscopyFramework` can save the spectroscopy game graph and formulas in Graphviz-format using `game.GameGraphVisualizer`, comparable to Figure 4.7. The output of the same mechanism for `spectroscopy.EquivalenceChecking` appears, for example, in the derived trace game of Figure 5.7.

Remark 8.1 (Line counting). When following the links to source listings above, you might notice that most aspects only take a few hundred lines of code. Depending on how one counts, a total of 2000–3000 lines fit the two spectroscopies, preorder checking for all the notions, game algorithms, generation and validation of HML formulas, and presentation mechanisms.

As far as code bases go, that size is extremely compact.¹³⁷ In part, this is enabled by Scala’s conciseness. But the heavy lifting in making the implementation so light-weight is achieved by the energy-game abstraction.

Remark 8.2 (Tests). The codebase of the backend of *equiv.io* includes test suites for strong and weak spectroscopy. These ensure that the algorithms (with and without optimizations) return the expected results for the finitary separating examples of strong and weak spectrum (van Glabbeek, 1990, 1993). This provides some confidence that not too much is going astray on the way between our correctness theorems and optimized implementation.

8.1.4 Benchmarks

Our algorithm and *equiv.io* are aimed at small transition systems that “fit onto one screen.” But still, the backend can analyze the equivalence structure of moderately-sized real-world transition systems. In this subsection, we examine its performance on the VLTS (“very large transition systems”) benchmark suite Garavel (2017) and on our recurring Peterson example.

Clever strong spectroscopy. Table 8.3 reports the results of running the backend with the clever strong spectroscopy game \mathcal{G}_Δ . This mostly matches the results already reported in Bisping (2023b).¹³⁸

The benchmark uses the VLTS examples of up to 25,000 states and the Peterson example. The table lists the \mathcal{P} -sizes of the input transition systems and of their strong bisimilarity quotient system \mathcal{P}/\sim_B (Definition 2.10).

The test suite constructs the game graph on the quotient system, starting at all positions that compare pairs of enabledness-equivalent states. The \rightarrow_Δ -column reports the size of the discovered game graph in terms of moves. The **time**-column lists execution times of the spectroscopy procedure in seconds.¹³⁹

¹³⁷ For reference, Cleaveland & Sims (1996) report 18,000 LOC for the *NSCU Concurrency Workbench* in Standard ML. Performance-centric C++ code bases like mCRL2 (Bunte et al., 2019; Groote & Mousavi, 2014) are another order of magnitude bigger.

¹³⁸ The benchmarking is in the class `tool.benchmark.VeryLargeTransitionSystems`. It can be started via `sbt "shared/run benchmark --strong-game --reduced-sizes --include-hard"`.

¹³⁹ Average of mid three runs out of five, in the Java Virtual Machine with 8 GB heap space, single-threaded on Intel® Core™ Ultra 7 155H.

The last three columns list the output sizes of state spaces reduced with respect to enabledness E, traces T, and simulation 1S.

From the output, we learn that the VLTS examples, in a way, lack diversity: Bisimilarity B and trace equivalence T coincide on the systems (third and penultimate column).

Concerning the algorithm itself, the experiments reveal that the computation time grows mostly linearly with the size of the game move graph. On `vasy_18_73`, the implementation times out after 500 seconds.

Of those terminating, the heavily nondeterministic `cwi_1_2` is the most expensive example. Almost all of its transitions are labeled by `i`, standing for internal activity τ in the VLTS suite. As many coarse notions must record the nondeterministic options, this blow-up is to be expected. If we compare to the best similarity algorithm by Ranzato & Tapparo (2010), they report their al-

Table 8.3: Runtime and results of strong spectroscopy on VLTS examples.

System	\mathcal{P}	\mathcal{P}/\sim_B	$\rightarrow_{\blacktriangle}$	time (s)	\mathcal{P}/\sim_E	\mathcal{P}/\sim_T	\mathcal{P}/\sim_{1S}
<code>peterson</code>	36	31	4,602	0.05	10	30	30
<code>vasy_0_1</code>	289	9	566	0.01	3	9	9
<code>cwi_1_2</code>	1,952	1,132	22,707,217	195.39	11	1,132	1,132
<code>vasy_1_4</code>	1,183	28	1,000	0.01	8	28	28
<code>cwi_3_14</code>	3,996	62	18,350	0.15	3	62	62
<code>vasy_5_9</code>	5,486	145	2,988	0.03	109	145	145
<code>vasy_8_24</code>	8,879	416	145,965	1.44	177	416	416
<code>vasy_8_38</code>	8,921	219	14,958	0.16	115	219	219
<code>vasy_10_56</code>	10,849	2,112	6,012,676	97.89	14	2,112	2,112
<code>vasy_18_73</code>	18,746	4,087	-	-	-	-	-
<code>vasy_25_25</code>	25,217	25,217	0	0.24	25,217	25,217	25,217

Table 8.4: Runtime and results of weak spectroscopy on VLTS examples.

System	\mathcal{P}	$\mathcal{P}/\sim_{BB^{\text{sr}}}$	\rightarrow_{∇}	time (s)	\mathcal{P}/\sim_{WE}	\mathcal{P}/\sim_{WT}	\mathcal{P}/\sim_{1WS}
<code>peterson</code>	36	21	3,579	0.10	5	16	16
<code>vasy_0_1</code>	289	9	2,958	0.07	3	9	9
<code>cwi_1_2</code>	1,952	67	22,944	0.40	19	67	67
<code>vasy_1_4</code>	1,183	4	0	0.00	4	4	4
<code>cwi_3_14</code>	3,996	2	0	0.00	2	2	2
<code>vasy_5_9</code>	5,486	112	2,723	0.06	91	112	112
<code>vasy_8_24</code>	8,879	170	10,983	0.12	115	169	169
<code>vasy_8_38</code>	8,921	193	32,598	0.35	104	193	193
<code>vasy_10_56</code>	10,849	2,112	-	-	-	-	-
<code>vasy_18_73</code>	18,746	2,326	-	-	-	-	-
<code>vasy_25_25</code>	25,217	25,217	0	0.32	25,217	25,217	25,217

gorithm SA to tackle `cwi_1_2` single-handedly. Like our implementation, the prototype of SA of Ranzato & Tapparo (2010) ran out of memory while determining similarity for `vasy_18_73`. This is in spite of SA theoretically having optimal complexity and similarity being less complex than trace equivalence, which we need to cover (cf. Section 3.3.3). The benchmarks in Ranzato & Tapparo (2010) failed at `vasy_10_56`, and `vasy_25_25`, which might be due to 2010's tighter memory requirements (they used 2 GB of RAM) or the degree to which bisimilarity and enabledness in the models is exploited.

Simplified weak spectroscopy. Table 8.4 lists analogous values for the weak spectroscopy game \mathcal{G}_{∇} , in the simplified variant, but also containing revivals moves of Section 7.3.2.

The algorithmic setup is slightly changed for the weak setting: We work with the quotient system of *stability-respecting branching bisimilarity* and start at *weakly* enabledness-equivalent pairs. The last three columns give the output quotient sizes for weak enabledness WE, weak traces WT, and weak simulation 1WS.

The peterson example, after minimization, has exactly the 21 states shown in Figure 1.2. Its weak similarity quotient is smaller (16 states, cf. last column), which fits our observations that weak similarity equates more in this model than bisimilarity-like notions do.¹⁴⁰

In weak semantics, `cwi_1_2` becomes easy, as its internal nondeterminism is directly compressed away by the initial branching-bisimilarity minimization.

Again, we see that branching bisimilarity and weak trace equivalence mostly coincide on the VLTS examples. Only `vasy_8_24` differs between 170 states in $\mathcal{P}/\sim_{\text{BB}^{\text{sr}}}$ and 169 in $\mathcal{P}/\sim_{\text{WT}}$.

The numbers align with the output of a different implementation in Bisping & Nestmann (2019), which is a good sign for the correctness of both programs. There, the same samples were analyzed with respect to coupled similarity, a weak notion close to contrasimilarity and weak similarity. Interestingly, the coupled simulation implementation in Bisping & Nestmann (2019) takes minutes for `vasy_25_25`, constructing a game with 126,000 moves. The trivial game of Table 8.4 suggests that this would not be necessary.

In summary, the equivalences of our spectra mostly coincide on the considered VLTS samples. This indicates that the examples are based around models that avoid the expressive power of finer branching-time notions.

¹⁴⁰ However, quotient systems need not be minimal for simulation-like and trace-like notions. For discussions on weak minimizations of Peterson's mutex, see Bouali (1992) and van Glabbeek & Weijland (1996).

8.2 Student Implementations

There exist three other implementations of the equivalence spectroscopy algorithm by students of mine. Each covers a different direction: In Section 8.2.1, we discuss *The Spectroscopy Invaders*, a computer-game version, which has an educational purpose. Section 8.2.2 presents an extension to an existing educational tool, the Concurrency Workbench Aalborg Edition. Section 8.2.3 closes

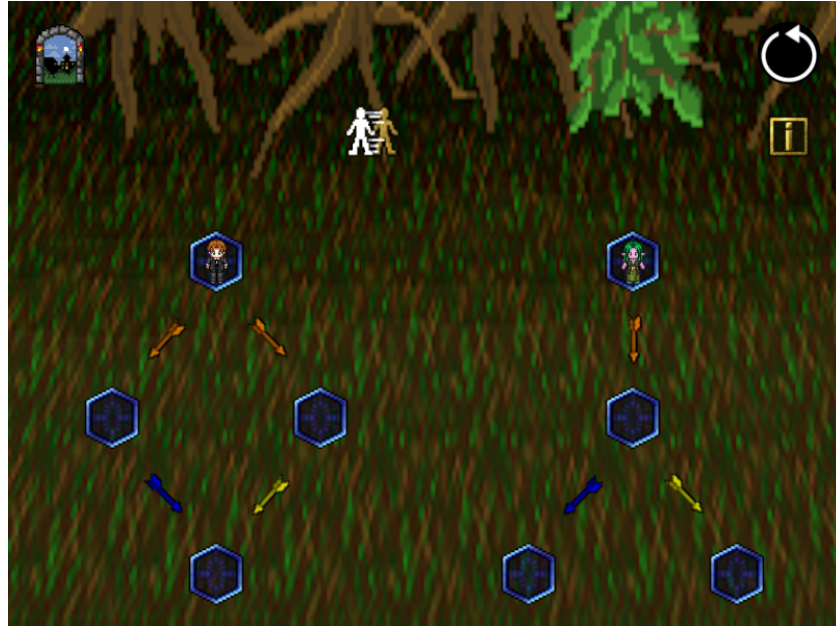


Figure 8.4: Screenshot of browser game “The Spectroscopy Invaders.”

by reporting on *gpuequiv*, a performance-centric shader-based implementation using the modern WebGPU standard.

8.2.1 Computer Game: *The Spectroscopy Invaders*

Would it not be nice if one could *play* the spectroscopy game *as a game*?

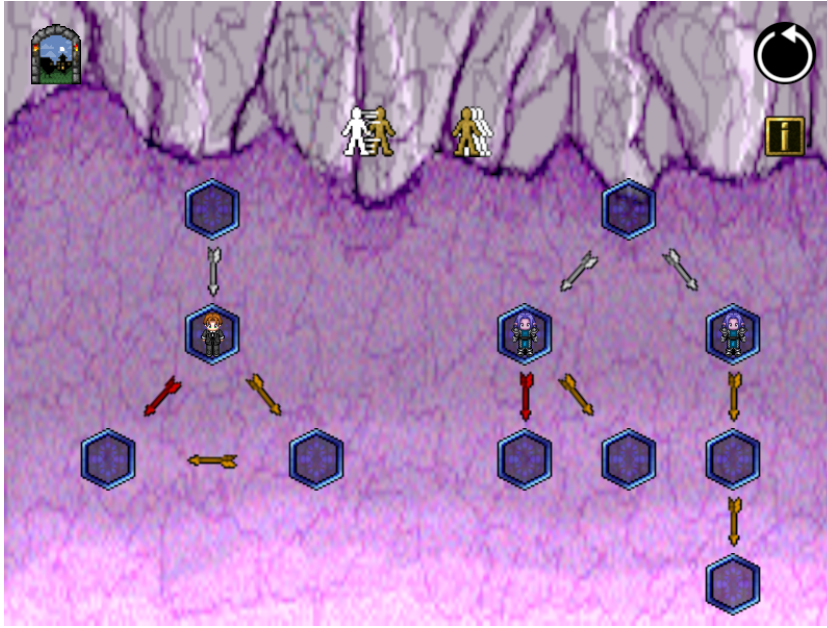
Trzeciakiewicz (2021) develops the computer game “The Spectroscopy Invaders” where one plays the attacker in the spectroscopy game. You can enjoy the game in the browser at <https://concurrency-theory.org/lbt-game/>.


A play of the game corresponds to naming a universal attack strategy in the strong spectroscopy game of Definition 5.1, or, equivalently, to constructing a distinguishing formula. To reach maximal level scores, one has to play out a minimal formula in the sense of our pricing. Under the hood, the TypeScript implementation uses Bisping et al.’s (2022) algorithm to compute the minimal budgets. The game does not present the costs of moves to the player, but they matter for the scores received at the end of a level.

Figure 8.4 gives a screenshot of the first level of the “Failure” world.¹⁴¹ This level corresponds to the classic Example 2.1 of P and Q if we match the orange transitions to τ -steps, blue to a, and yellow to b. The metaphor on top of the game mechanics is that the human-controlled hero (left) has to outmaneuver a group of elves (right) that might split up on nondeterministic transitions.

From Example 5.3, we know that the failure $\langle \tau \rangle \neg \langle a \rangle$ is a cheapest distinction of the left state from the right in this system, translated as $\langle \text{orange} \rangle \neg \langle \text{blue} \rangle$. In the computer game, we would use this distinction as


¹⁴¹ In the online version of this thesis, it is actually a screen capture by Trzeciakiewicz, illustrating negation moves in the game tutorial.

Figure 8.5: Second level of *Simulation* world.

follows: First click on the state behind the middle orange transition to *observe* it. Then, click on the “negation” button  to swap sides of hero and elve. Defeat them by moving along the right blue edge.

The handling of conjunctions is a little more involved.

Figure 8.5 shows the second level of the “Simulation” world. In the level, the player’s task is to distinguish $\text{white} \cdot (\text{red} + \text{orange} \cdot \text{orange})$ (left) from $\text{white} \cdot (\text{red} + \text{orange}) + \text{white} \cdot \text{orange} \cdot \text{orange}$ (right). The current game position occurs after the first white observation and corresponds to $[\text{red} + \text{orange} \cdot \text{orange}, \{\text{red} + \text{orange}, \text{orange} \cdot \text{orange}\}]_a$ in the strong spectroscopy game \mathcal{G}_Δ , as indicated by the hero figure on the left and the two elves on the right.

The player can click the “conjunction” button , after which they have to explain how to win $[\text{red} + \text{orange} \cdot \text{orange}, \text{red} + \text{orange}]_a^\wedge$ and then $[\text{red} + \text{orange} \cdot \text{orange}, \text{orange} \cdot \text{orange}]_a^\wedge$. In the split state, the player can first click through the yellow-arrow successor states on the left to point out that the left elve cannot observe two yellow steps. Then, they also have to name how to defeat the right elve, by taking the red transition. Together, this strategy corresponds to the simulation formula $\bigwedge \{ \langle \text{yellow} \rangle \langle \text{yellow} \rangle, \langle \text{red} \rangle \}$.

The game is single-player, also in a theoretical sense: There is no picking of conjunction answers by the defender. Instead, the attacker has to name attacks for every right-hand state. Due to nested conjunctions, the game positions thus actually are sets of $[p, Q]_a$ tuples.

Trzeciakiewicz (2021) limits the scope to (strong) trace, failure, possible-future, simulation, and bisimulation equivalences, excluding notions like

readiness, ready traces, and failure traces. The selection permits slightly simpler game rules—in particular, without revival moves. This allows to nicely showcase the core mechanics of Chapter 5 with a stricter correspondence to productions in the original HML of Definition 2.11.

8.2.2 CAAL Extension

Would it not be nice if one could use the equivalence spectroscopy in *existing tools*?

Ozegowski (2023) integrates the spectroscopy algorithm into *CAAL*, the “Concurrency Workbench Aalborg Edition” by Andersen et al. (2015). Straßnick & Ozegowski (2024) also add the weak spectroscopy game and possibilities to play the spectroscopy game in the tool. Their extended version of CAAL is live on <https://equivio.github.io/CAAL/>.

The original CAAL (on <https://caal.cs.aau.dk/>) covers the curriculum of *Reactive systems* (Aceto et al., 2007), including CCS processes, HML formulas and equivalence games.

Originally, CAAL only supports preorder/equivalence checking for six standard notions, namely for simulation, bisimulation, and traces, in their standard strong and weak variants. Only simulation and bisimulation can be examined as games.

Straßnick and Ozegowski’s extended CAAL version supports 13 strong and 21 weak notions. Each of the notions can be decided individually, or in the context of a spectroscopy. For the strong notions, the game graph can be explored interactively. Figure 8.6 shows the output of strong and weak spectroscopy on the classic Example 2.1, together with a generated distinguishing failure formula in the HML dialect of CAAL.

For a usage guide, we refer to Straßnick & Ozegowski (2024).

At some points, Straßnick and Ozegowski’s fork unfortunately has to remain partial with respect to features. For instance, the extension does not support distinguishing formulas for the weak spectroscopy because CAAL’s HML dialect cannot easily be molded to support HML_{SRBB} .

Still, the project shows that the spectroscopy approach is sufficiently simple and versatile to allow dozens of equivalence checkers to be integrated into existing tools within the limited scope of a student project.

8.2.3 GPU Implementation: *gpuequiv*

Would it not be nice if one could use modern hardware to perform spectroscopies as fast as possible?

Vogel (2024) implements the strong equivalence spectroscopy in *gpuequiv* using shaders in the WebGPU Shading Language (Baker et al., 2025).

Technically, this is superior to the other three implementations of this chapter, which can only exploit the single-threaded CPU model of JavaScript when running in the browser.

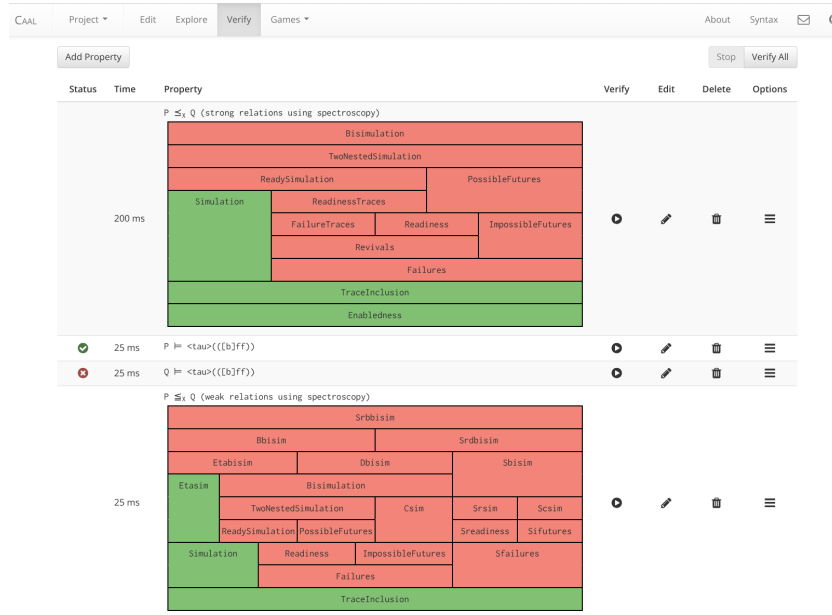


Figure 8.6: Extended version of CAAL, performing a spectroscopy.

Historically, the WebGL standard for browser graphics processing has lacked compute shaders. This has made it difficult to access the computational power of graphics processing units from within web applications. Times are changing with WebGPU/WGSL (Baker et al., 2025). Vogel (2024) makes this technological progress available to the spectroscopy approach.

Big parts of the game graph construction and the game solving are quite parallelizable. `gpuequiv` parallelizes the budget computation of Algorithm 4.1. For instance, a whole batch of game positions on a todo-list can be processed simultaneously. The details are explained by Vogel (2024).

`gpuequiv`'s control code surrounding the shader invocations is written in Rust that can be compiled to native code and to WebAssembly. Therefore, `gpuequiv` can be compiled for both kinds of targets: Quick equivalence checking in the browser, and even quicker checking in a native application. Vogel (2024) reports $10\times$ speed-ups compared to Bisping (2023b). However, the fast growth of game graphs remains a limiting factor as larger examples run into the size boundary of buffers to upload game moves. For the system `vasy_18_73`, Vogel (2024) also fails to complete the spectroscopy (like our experience in Section 8.1.4): It constructs a game graph of 623,482,227 moves, but only 536,870,911 moves fit into the buffer with the employed data packing.

At the time of writing, another Bachelor project is underway to add a frontend and support for the *weak* spectroscopy game to `gpuequiv`.

8.3 Discussion

In this chapter, we have surveyed the four existing implementations of the spectroscopy algorithm.

We focused on [equiv.io](#), which closely aligns with this thesis. We have seen how the tool easily answers questions such as “How equivalent is Peterson’s algorithm to the specification of mutual exclusion.” [equiv.io](#) can check behavioral preorders and equivalences for more than 38 notions of strong and weak spectrum. Due to the possibility to mark divergences, even more notions are available via preprocessing.

Other tools. A recent survey by Garavel & Lang (2022) lists hundreds of tools to check bisimilarity and related equivalences and preorders. Some of them can justify their output with distinguishing formulas (or traces), similar to our approach. Many specific tools address special equivalences, for instance for open, timed, or probabilistic systems, which we do not support. On the other hand, [equiv.io](#) might be the first tool to cover some of the more arcane notions of the weak spectrum (van Glabbeek, 1993), such as η -similarity and stable bisimilarity.

In Table 8.5, we compare the supported notions of [equiv.io](#) to four current state-of-the-art tools. A tick ✓ stands for direct support, (✓) for support via preprocessing.

- CAAL (Andersen, Andersen, et al., 2015), or the “Concurrency Workbench Aalborg Edition,” also works with CCS and has already been discussed in Section 8.2.2.¹⁴²
- mCRL2 (Groote & Mousavi, 2014), built around an ACP/CCS-like modelling language of the same name, supports a wide-range of notions with highly efficient implementations and a strong focus on branching bisimilarity. On the notions it supports, it will generally be much faster than the spectroscopy game algorithm. The only notion that is exclusively supported by mCRL2 is coupled similarity, implemented by Lê (2020).
- CADP (Garavel et al., 2013) can check several notions on-the-fly, also in a highly optimized fashion. Its models are usually expressed in LOTOS or LNT, deriving from CSP. It includes the special notions of safety- and $\tau^*.a$ -equivalence, not present in van Glabbeek’s spectrum (1993).
- FDR4 (Gibson-Robinson et al., 2014) has “Failures Divergences Refinement” in its name, but also supports different linear-time refinement models for CSP. Branching-time notions are partially supported as minimizers.

Like the listed tools, [equiv.io](#) can be seen as following the tradition of the discontinued “Concurrency Workbench[es]” (Cleaveland et al., 1990; Cleaveland & Sims, 1996).

¹⁴² Andersen, Hansen, et al. (2015, Section 3.3) also mention how to use their game backend to handle 2-nested simulation and ready-traces. But this is neither elaborated upon nor supported in the frontend.

Other avenues for implementation. The implementations of this chapter realize Algorithm 4.1 quite directly to decide energy games. Among them, the shader version of Section 8.2.3 adds the highest level of implementation cleverness with respect to packing of data and parallelization of computation. But so far, we are not using the full toolbox of the computer-aided-verification community.

For instance, the upward-closed sets of winning budgets could be handled more symbolically with the representation of Delzanno & Raskin (2000).

Table 8.5: Comparison of supported notions.

Equivalence / preorder	equiv.io	CAAL	mCRL2	CADP	FDR4
Strong / weak enabledness	✓				
Strong trace	✓	✓	✓	✓	
Weak trace	✓	✓	✓	✓	✓
Strong failure	✓		✓		
Weak failure	✓				
Stable failure	✓		✓		✓
Failure-divergence	(✓)		✓		✓
Strong / stable revivals	✓				
Strong / weak / stable readiness	✓				
Strong / stable failure-trace	✓				
Strong / stable ready-trace	✓				
Strong / stable impossible fut.	✓				
Weak impossible future	✓		✓		
Strong / weak possible future	✓				
Strong simulation	✓	✓	✓		
Weak simulation	✓	✓			
Stable simulation	✓				
η -simulation	✓				
Safety / $\tau^*.a$				✓	
Strong ready-simulation	✓		✓		
Weak / stable ready-simulation	✓				
2-nested strong / weak sim.	✓				
Strong bisimulation	✓	✓	✓	✓	(✓)
Contrasim. / stable bisim.	✓				
Coupled simulation			✓		
Weak bisimulation	✓	✓	✓	✓	(✓)
Div.-pr. weak bisim.	(✓)		✓		(✓)
Delay bisimulation	✓				(✓)
Stability-resp. delay bisim.	✓				
η -bisimulation	✓				
Branching bisimulation	✓		✓	✓	
Stability-resp. branch. bisim.	✓				
Div.-pr. branching bisim.	(✓)		✓		(✓)

This might slightly increase how many Pareto fronts can be kept in memory. But one can expect that its payoff would be meager, especially on the small $\{0, 1, \infty\}^d$ -lattices. Experiments by Vogel (2024) suggest that the Pareto fronts usually stay sufficiently small that adding the implementation complexity of Delzanno & Raskin (2000) seems uncalled for.

A big limiting factor of our implementations is the storage of the game graph. To some extent, this can be battled by more symbolic representations as *binary-decision-diagrams*. Bulychev (2011) and Wimmer (2011) follow this route in quite versatile checkers.

A more general solution for the memory limitations would be to forget parts that have been visited, and recompute them *by-need*. There are easy ways to profit from the community’s advances in handling big spaces of possibilities. The most prominent would be to instantiate the game rules for a transition system, and feed them into an SMT solver. Already Shukla et al. (1996) suggest a comparable approach, viewing games as SAT problems. The energy aspect should be perfectly expressible for SMT/SAT solvers in linear integer arithmetic (cf. Chistikov, 2024).

Alternative paths to generalized checkers. In Section 2.5, we have already discussed the alternative paradigms of equivalence checking. The *equivalences-as-game-instances* approach seems to be the most fruitful when one wants to easily support a wide range of preorders and equivalences in a tool. By “easy,” we mean: without the need to implement individual algorithms for each of the notions. Tan (2002a) takes a similar game approach for the Concurrency Workbench, as do Andersen, Hansen, et al. (2015) for CAAL.

Another route to this goal of genericity might be the approach of *equivalences-as-formulas* from Lange et al. (2014). The idea there is to use a *diadic higher-order μ -calculus* where one can talk about relations between states. Thereby, the rules of how to preorder states can be expressed as formulas. One only needs to implement the model-checking for formulas once, and new notions could be added by only adding new formulas. Stöcker (2024) implements this idea successfully. But the data suggests that 20 states already demand several seconds in this approach for some equivalences.

The third path to genericity is through *equivalences-as-functors*, to enable *coalgebraic partition-refinement algorithms* from category theory. We already hinted to this in Section 2.5. Deifel et al. (2019) follow this route in their tool *CoPaR* (short for “Coalgebraic Partition Refiner”). This way, the support of new equivalences boils down to a few lines of defining new functors. The big advantage is that extensions—for instance, to quantitative behavioral distances—can also be achieved this way. The coalgebraic approach is related to the game approach and, too, can derive distinguishing formulas, as seen for instance in the tool T-BEG (König et al., 2020). However, encoding simulation-like preorders in category theory seems to be non-trivial if one compares the machinery of Ranzato & Tapparo (2010) to the ease of just leav-

ing out swap-moves in the bisimulation game. It is difficult to say whether it is conceptional boundaries or the coating in category-theory parlance that has hindered a wider adoption in tools.

A light-weight variant of the functor-approach is offered by *equivalences-as-signatures*. Tools like *Sigref* (Wimmer et al., 2006) support a broad range of bisimulation-like equivalences. The specifics of individual equivalences are expressed as *signatures* that prescribe how to refine equivalence classes in an iterative partition-refinement. Signature Refinement can easily be boosted by parallelization or symbolic BDD-encodings.

Of course, all these theoretical approaches are linked on some basic level, namely through the Hennessy–Milner theorem (Theorem 2.1). We elaborate on this in the conclusion.

9 Conclusion

The main contribution of this thesis has been to solve Problem 1, the *spectroscopy problem*, deciding all behavioral preorders and equivalences of strong and weak spectrum, in one go. For this objective, we have provided *generalized game characterizations* for both spectra. To turn these into decision procedures, we have developed a theory of *Galois energy games*. Thanks to the decision procedures, we can offer *tool support* for equivalence spectroscopies. So, indeed, there is such a thing as “linear-time–branching-time spectroscopy” (Idea 1).

The big picture. Figure 9.1 shows the steps our theory has taken:

- We have started at the *level of individual equivalences* by revisiting the classical Hennessy–Milner theorem on the dualism of modal distinguishability and behavioral relatibility for bisimilarity.
- Chapter 2 has revealed how relations and modal formulas in the Hennessy–Milner theorem are just *certificates* for defender and attacker strategies in the bisimulation reachability game (Idea 2).
- Chapter 3 has taken us to the level of *spectra* of equivalence, following van Glabbeek’s approach of a *hierarchy of Hennessy–Milner theorems* (Idea 3, Idea 4). The spectroscopy problem (Idea 5) emerges naturally.
- On this level, Chapter 4 has discovered that the bisimulation game just is an instance of the bisimulation *energy game*, which characterizes a small spectrum of P-easy strong equivalences (Idea 6). We can use it

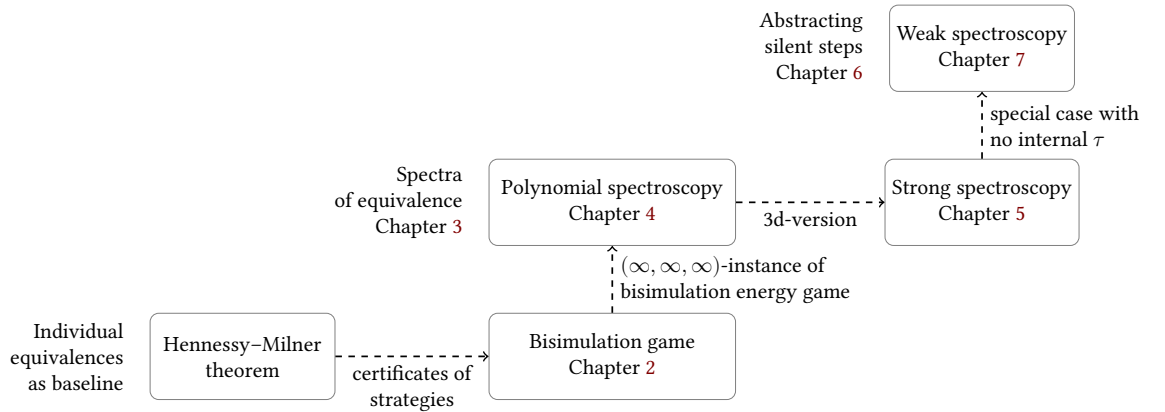


Figure 9.1: Hierarchy of abstraction from Hennessy–Milner theorem and strong bisimilarity to weak spectroscopy. (Read arrows as “... can be understood as ... of ...”.)

in algorithms thanks to our decision procedure for such *Galois energy games* (Idea 7).

- The bisimulation energy game then again is only a lower-dimensional sibling of the *strong spectroscopy game*, which characterizes the full strong spectrum in Chapter 5 by a subset construction (Idea 8). We can still project it back all the way to the bisimulation game (Idea 9).
- With Chapter 6, we have entered the level of van Glabbeek’s (1993) *weak spectrum*, where silent steps are abstracted, through a weakening of HML (Idea 10).
- Chapter 7 has introduced the *weak spectroscopy game*, where the defender profits from internal transitions (Idea 11). Where there is no internal τ -activity, the game could still serve as a strong spectroscopy game.

In much the same way as we understand equivalences in a hierarchy by folding away dimensions of distinctions, this thesis itself has followed a hierarchy of spectroscopies. Starting from the Hennessy–Milner theorem, we have *unfolded* aspects, up to the weak spectroscopy. If one looks closely, we have even started one level below: Remark 2.6 notices that the subtractive \leq -game on natural numbers is an instance of the simulation game.

One could frame this journey along Plato’s allegory of the cave: The Hennessy–Milner theorem *is a shadow* of the bisimulation game, which is a shadow of an energy game, which is a shadow of the strong spectroscopy game, which is a shadow of the weak spectroscopy game. With the tool implementations of Chapter 8, we have returned from the abstract realm of ideas back to a more material level.

But in reality, there is not one single *ideal* hierarchy of abstraction. Our journey has crossed many routes by other researchers and could have taken different paths. Many interesting ones have been the topic of “Discussion” sections at the end of chapters. Let us close by looking at some ways one could go from here.

Blank spots. There are some classical points to extend research around this thesis. For instance:

- More formalization: Our Isabelle/HOL theory of the weak spectroscopy strengthens many results of the thesis (thanks to the hierarchy of shadows, Figure 9.1). But so far, it does not cover revivals and thus failure-trace-like notions. The proofs in Bisping (2023c) give some confidence that the construction is correct, but an Isabelle/HOL formalization would be even nicer.
- More Galois energy games: There exist some interesting questions around the theory of Galois energy games of Chapter 4 and their generalizability to more game variants.¹⁴³
- More tool support: Proliferation of spectroscopy features to more tools

¹⁴³ Lemke and Bisping are already writing a paper on this.

beyond Chapter 8 would be great. Our existing open source implementations can hopefully serve as good starting points.

More spectra. Concurrency theory knows other variants of linear-time–branching-time spectra. For some, it might be worthwhile to ask spectroscopy questions or to adapt our approach of generalized game characterizations.

- There are various *quantitative notions*, often understood in terms of *behavioral distances*. They could be included into our game model by allowing the defender a certain amount of “cheating.” For instance, the Q -sets in game positions could be fuzzy, and some budget could define how unlikely positions the defender can pick.

The (strong) quantitative spectrum of Fahrenberg & Legay (2014) already is phrased in terms of a game model. Combining it with our game would likely demand deeper changes to our game mechanics, because one has to remember traces (and not just current states) to compute the resulting distances in full generality.

For Markovian models and probabilistic bisimilarities, there is recent work of “spectro-fication” in Bian & Abate (2017), Baier et al. (2020), and Spork et al. (2024), also extending to silent-step notions. So, this might be the kind of quantitative equivalences where a connection to our approach works best.

Timed equivalences would face conceptual boundaries to decision procedures on finite models: While bisimilarity is decidable, trace pre-ordering is not (Čerāns, 1993).

- *Barbed and other congruences* on π -calculi rely on some system of *contexts* in which to place processes. Fournet & Gonthier (2005) present a hierarchy for asynchronous name-passing calculi, on which one could base a spectroscopy. Valmari (2020) even identifies a spectrum of *all* linear-time congruences for a sensible set of operators.

Conceptually, it is clear that the selection of a distinguishing context could also happen as an attacker move. In practice, it might be hard to do this elegantly in a way where games remain finite for finite systems. Likely, the most feasible route leads through *conditional equivalences* (Hülsbusch et al., 2020).

- Some kinds of bisimilarity are based on *special modalities*. For instance, there are forward/reverse bisimilarities (Bernardo & Esposito, 2023) that add modalities to move backwards through the transition graph.

Many such modalities could be added as special moves to the game graph. For modalities where satisfaction depends on infinite behavior, the reachability game framework is not sufficiently expressive. An example would be the divergence modalities mentioned in Section 7.3.3, which require a parity-game winning condition.

A modal-logical turn? This thesis has approached the rich body of work on equivalence and refinement through one unified lens of *finding differences* in behavior. These were expressed in *modal logics* (Idea 3 “Modal first!”). The *perks of modal characterizations* have been a recurring topic of the thesis.

In recent years, there seems to be a shift towards the “modal first” approach. The shift is visible in Geuvers and Golov’s work on apartness (Geuvers, 2022; Geuvers & Golov, 2023), similarly in Ford et al. (2021) and Forster et al. (2024) connecting graded modal logics and monads, in Wißmann et al. (2021) strongly linking partition refinement and modal logic, as well as in Beohar et al. (2023) using Galois connections to obtain Hennessy–Milner theorems.

This thesis and its surrounding publications, too, participate in this movement. However, we leave the stack of abstractions around coalgebras and monads aside. One of the effects of this decision is that our framework can be implemented by regular Bachelor students after one introductory course on reactive systems.

The encyclopedic project. Concurrency theory as a field has entered a stage where the initial explosion of questions and approaches has faded. A wave of Festschriften has been appearing. As Garavel (2023) puts it:

Time has come for encyclopedic synthesis: reexamine / select /
simplify / sort

This thesis hopes to fit into this project with a “search engine” for equivalences. It shows how to quickly survey spectra of equivalence and how the approach can easily be implemented into tools to support generalized equivalence checking of concurrent programs. While pioneers of the field are retiring, parts of their wisdom thus can be materialized into new generations of software.

After all, concurrent computation will remain relevant, and so will questions on the equivalence of concurrent programs.

References

- Aceto, L., Fokkink, W., van Glabbeek, R., & Ingólfssdóttir, A. (2004). Nested semantics over finite trees are equationally hard. *Information and Computation*, 191(2), 203–232. <https://doi.org/10.1016/j.ic.2004.02.001>
- Aceto, L., Ingólfssdóttir, A., Larsen, K. G., & Srba, J. (2007). *Reactive systems: Modelling, specification and verification* (pp. 142–158). Cambridge University Press. <https://doi.org/10.1017/CBO9780511814105.008>
- Adler, R. (2022). *Simulation fehlertoleranter Konsensalgorithmen in Hash.ai* [Bachelor's thesis]. Technische Universität Berlin.
- Allaire, J. J., Teague, C., Scheidegger, C., Xie, Y., Dervieux, C., & Woodhull, G. (2025). *Quarto* (Version 1.7) [Computer software]. <https://doi.org/10.5281/zenodo.5960048>
- Alshukairi, Z. (2022). *Automatisierte Reduktion von reaktiver zu starker Bisimilarität* [Bachelor's thesis]. Technische Universität Berlin.
- Andersen, J. R., Andersen, N., Enevoldsen, S., Hansen, M. M., Larsen, K. G., Olesen, S. R., Srba, J., & Wortmann, J. K. (2015). CAAL: Concurrency workbench, aalborg edition. In M. Leucker, C. Rueda, & F. D. Valencia (Eds.), *Theoretical aspects of computing — ICTAC 2015* (pp. 573–582). Springer International Publishing. https://doi.org/10.1007/978-3-319-25150-9_33
- Andersen, J. R., Hansen, M. M., & Andersen, N. (2015). *CAAL 2.0 – equivalences, preorders and games for CCS and TCCS* (Vol. DES104F15) [Student project supervised by Jiří Srba and Kim Guldstrand Larsen]. Aalborg University. https://caal.cs.aau.dk/docs/CAAL2_EPG.pdf
- Babai, L. (2016). Graph isomorphism in quasipolynomial time [extended abstract]. *STOC '16*, 684–697. <https://doi.org/10.1145/2897518.2897542>
- Baier, C., D'Argenio, P. R., & Hermanns, H. (2020). On the probabilistic bisimulation spectrum with silent moves. *Acta Informatica*, 57(3), 465–512. <https://doi.org/10.1007/s00236-020-00379-2>
- Baker, A., Derin, M. O., Neto, D., Maxfield, M. C., & Sinclair, D. (Eds.). (2025). *WebGPU shading language* (Version W3C Recommendation Draft). World Wide Web Consortium. <https://www.w3.org/TR/WGSL/>
- Balcázar, J., Gabarro, J., & Santha, M. (1992). Deciding bisimilarity is P-complete. *Formal Aspects of Computing*, 4, 638–648. <https://doi.org/10.1007/BF03180566>
- Barthel, L. A., Bisping, B., Hübner, L. M., Lemke, C., Mattes, K. P. P., & Mollenkopf, L. (2025). *A weak spectroscopy game to characterize behavioral equivalences* [Isabelle/HOL formalization]. https://equivio.github.io/silent-step-spectroscopy/AFP/Weak_Spectroscopy.
- Barwell, A. D., Ferreira, F., & Yoshida, N. (2022). CONCUR test-of-time award

- for the period 1994–97 interview with Uwe Nestmann and Benjamin C. Pierce. *Journal of Logical and Algebraic Methods in Programming*, 125, 100744. <https://doi.org/10.1016/j.jlamp.2021.100744>
- Basten, T. (1996). Branching bisimilarity is an equivalence indeed! *Information Processing Letters*, 58(3), 141–147. [https://doi.org/10.1016/0020-0190\(96\)00034-8](https://doi.org/10.1016/0020-0190(96)00034-8)
- Bell, C. J. (2013). Certifiably sound parallelizing transformations. In *International conference on certified programs and proofs* (pp. 227–242). Springer. https://doi.org/10.1007/978-3-319-03545-1_15
- Bell, C. J. (2014). *A proof theory for loop-parallelizing transformations* [PhD thesis, Princeton University]. <https://people.csail.mit.edu/cj/thesis/thesis.pdf>
- Beohar, H., Gurke, S., König, B., & Messing, K. (2023). Hennessy-Milner Theorems via Galois Connections. In B. Klin & E. Pimentel (Eds.), *LIPICs: Vol. 252. 31st EACSL annual conference on computer science logic (CSL 2023)* (pp. 12:1–12:18). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. <https://doi.org/10.4230/LIPICs.CSL.2023.12>
- Bernardo, M., & Esposito, A. (2023). Modal logic characterizations of forward, reverse, and forward-reverse bisimilarities. *EPTCS*, 390, 67–81. <https://doi.org/10.4204/eptcs.390.5>
- Bian, G., & Abate, A. (2017). On the relationship between bisimulation and trace equivalence in an approximate probabilistic context. In J. Esparza & A. S. Murawski (Eds.), *Foundations of software science and computation structures* (pp. 321–337). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-54458-7_19
- Bisping, B. (2018). *Computing coupled similarity* [Master’s thesis, Technische Universität Berlin]. https://coupledsim.bisping.de/bisping_computingCoupledSimilarity_thesis.pdf
- Bisping, B. (2021). *Linear-time–branching-time spectroscopy: TACAS 2021 edition* [Artifact archived on Zenodo]. Zenodo. <https://doi.org/10.5281/zenodo.4475878>
- Bisping, B. (2023a). *Linear-time–branching-time spectroscopy v0.3.0* (Version v0.3.0) [Artifact archived on Zenodo]. Zenodo. <https://doi.org/10.5281/zenodo.7870252>
- Bisping, B. (2023b). Process equivalence problems as energy games. In C. Enea & A. Lal (Eds.), *Computer aided verification* (pp. 85–106). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-37706-8_5
- Bisping, B. (2023c). *Process equivalence problems as energy games* [Technical Report]. Technische Universität Berlin. <https://doi.org/10.48550/arXiv.2303.08904>
- Bisping, B., Brodmann, P.-D., Jungnickel, T., Rickmann, C., Seidler, H., Stüber, A., Wilhelm-Weidner, A., Peters, K., & Nestmann, U. (2016). Mechanical verification of a constructive proof for FLP. In J. C. Blanchette & S. Merz (Eds.), *LNCS: Vol. 9807. Interactive theorem proving* (pp. 107–122). Springer International Publishing.

- https://doi.org/10.1007/978-3-319-43144-4_7
- Bisping, B., & Jansen, D. N. (2024). One energy game for the spectrum between branching bisimilarity and weak trace semantics. In G. Caltais & C. Di Giusto (Eds.), *EPTCS: Vol. 412. Proceedings Combined 31st International Workshop on expressiveness in concurrency and 21st Workshop on structural operational semantics, Calgary, Canada, 9th September 2024* (pp. 71–88). Open Publishing Association. <https://doi.org/10.4204/EPTCS.412.6>
- Bisping, B., & Jansen, D. N. (2025). *Deciding all behavioral equivalences at once II: The silent-step spectrum* [Preprint]. https://code.keks.in/papers/2025/silent-step-spectroscopy_journal.pdf
- Bisping, B., Jansen, D. N., & Nestmann, U. (2022). Deciding all behavioral equivalences at once: A game for linear-time–branching-time spectroscopy. *Logical Methods in Computer Science*, 18(3). [https://doi.org/10.46298/lmcs-18\(3:19\)2022](https://doi.org/10.46298/lmcs-18(3:19)2022)
- Bisping, B., & Montanari, L. (2021). A game characterization for contrasimilarity. In O. Dardha & V. Castiglioni (Eds.), *EPTCS: Vol. 339. Proceedings Combined 28th International Workshop on expressiveness in concurrency and 18th Workshop on structural operational semantics* (pp. 27–42). Open Publishing Association. <https://doi.org/10.4204/EPTCS.339.5>
- Bisping, B., & Montanari, L. (2023). Coupled similarity and contrasimilarity, and how to compute them. *Arch. Formal Proofs*, 2023. https://www.isa-afp.org/entries/CoupledSim_Contrasim.html
- Bisping, B., & Montanari, L. (2024). Characterizing contrasimilarity through games, modal logic, and complexity. *Inf. Comput.*, 300, 105191. <https://doi.org/10.1016/J.IC.2024.105191>
- Bisping, B., & Nestmann, U. (2019). Computing coupled similarity. In T. Vojnar & L. Zhang (Eds.), *LNCS: Vol. 11427. Tools and algorithms for the construction and analysis of systems: TACAS* (pp. 244–261). Springer. https://doi.org/10.1007/978-3-030-17462-0_14
- Bisping, B., & Nestmann, U. (2021). A game for linear-time–branching-time spectroscopy. In J. F. Groote & K. G. Larsen (Eds.), *LNCS: Vol. 12651. Tools and algorithms for the construction and analysis of systems: TACAS* (pp. 3–19). Springer. https://doi.org/10.1007/978-3-030-72016-2_1
- Bisping, B., Nestmann, U., & Peters, K. (2020). Coupled similarity: The first 32 years. *Acta Informatica*, 57(3–5), 439–463. <https://doi.org/10.1007/s00236-019-00356-4>
- Bouali, A. (1992). *Weak and branching bisimulation in FCTOOL*. INRIA. <https://inria.hal.science/inria-00074985>
- Brookes, S. D., Hoare, C. A. R., & Roscoe, A. W. (1984). A theory of communicating sequential processes. *J. ACM*, 31(3), 560–599. <https://doi.org/10.1145/828.833>
- Bulik, J. (2021). *Statically analysing inter-process communication in Elixir programs* [Bachelor's thesis, Technische Universität Berlin]. <https://gitlab.com/MrGreenTea/stille-post>
- Bulychev, P. E. (2011). Game-theoretic simulation checking tool. *Pro-*

- gramming and Computer Software*, 37(4), 200. <https://doi.org/10.1134/S0361768811040013>
- Bunte, O., Groote, J. F., Keiren, J. J. A., Laveaux, M., Neele, T., Vink, E. P. de, Wesselink, W., Wijs, A., & Willemse, T. A. C. (2019). The mCRL2 toolset for analysing concurrent systems. In T. Vojnar & L. Zhang (Eds.), *Tools and algorithms for the construction and analysis of systems* (pp. 21–39). Springer International Publishing. https://doi.org/10.1007/978-3-030-17465-1_2
- Čerāns, K. (1993). Decidability of bisimulation equivalences for parallel timer processes. In G. von Bochmann & D. K. Probst (Eds.), *Computer aided verification* (pp. 302–315). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-56496-9_24
- Chen, X., & Deng, Y. (2008). Game characterizations of process equivalences. In G. Ramalingam (Ed.), *LNCS: Vol. 5356. Programming languages and systems: APLAS* (pp. 107–121). Springer. https://doi.org/10.1007/978-3-540-89330-1_8
- Chistikov, D. (2024). An Introduction to the Theory of Linear Integer Arithmetic. In S. Barman & S. Lasota (Eds.), *LIPIcs: Vol. 323. 44th IARCS annual conference on foundations of software technology and theoretical computer science (FSTTCS 2024)* (pp. 1:1–1:36). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. <https://doi.org/10.4230/LIPIcs.FSTTCS.2024.1>
- Cleaveland, R. (1991). On automatically explaining bisimulation inequivalence. In E. M. Clarke & R. P. Kurshan (Eds.), *LNCS: Vol. 531. Computer-aided verification: CAV'90* (pp. 364–372). Springer. <https://doi.org/10.1007/BFb0023750>
- Cleaveland, R., & Hennessy, M. (1993). Testing equivalence as a bisimulation equivalence. *Formal Aspects of Computing*, 5(1), 1–20. <https://doi.org/10.1007/BF01211314>
- Cleaveland, R., Parrow, J., & Steffen, B. (1990). The concurrency workbench. In J. Sifakis (Ed.), *Automatic verification methods for finite state systems* (pp. 24–37). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-52148-8_3
- Cleaveland, R., & Sims, S. (1996). The NCSU concurrency workbench. In R. Alur & T. A. Henzinger (Eds.), *Computer aided verification* (pp. 394–397). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-61474-5_87
- Cleaveland, R., & Sokolsky, O. (2001). Equivalence and preorder checking for finite-state systems. In J. A. Bergstra, A. Ponse, & S. A. Smolka (Eds.), *Handbook of process algebra* (pp. 391–424). Elsevier Science. <https://doi.org/10.1016/B978-044482830-9/50024-2>
- De Nicola, R., & Vaandrager, F. (1995). Three logics for branching bisimulation. *J. ACM*, 42(2), 458–487. <https://doi.org/10.1145/201019.201032>
- Deifel, H.-P., Milius, S., Schröder, L., & Wißmann, T. (2019). Generic partition refinement and weighted tree automata. In M. H. ter Beek, A. McIver, & J. N. Oliveira (Eds.), *Formal methods – the next 30 years* (pp. 280–297). Springer International Publishing.

- https://doi.org/10.1007/978-3-030-30942-8_18
- Delzanno, G., & Raskin, J.-F. (2000). Symbolic representation of upward-closed sets. In S. Graf & M. Schwartzbach (Eds.), *Tools and algorithms for the construction and analysis of systems* (pp. 426–441). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-46419-0_29
- Duong, T. M. (2022). *Developing an educational tool for linear-time–branching-time spectroscopy* [Bachelor's thesis]. Technische Universität Berlin.
- England, J. (2021). *HML synthesis of distinguished processes* [Bachelor's thesis]. Technische Universität Berlin.
- Erné, M., Kosłowski, J., Melton, A., & Strecker, G. E. (1993). A primer on Galois connections. *Annals of the New York Academy of Sciences*, 704(1), 103–125. <https://doi.org/10.1111/j.1749-6632.1993.tb52513.x>
- Ésik, Z., Fahrenberg, U., Legay, A., & Quaas, K. (2013). Kleene algebras and semimodules for energy problems. In D. Van Hung & M. Ogawa (Eds.), *Automated technology for verification and analysis* (pp. 102–117). Springer International Publishing. https://doi.org/10.1007/978-3-319-02444-8_9
- Fahrenberg, U., Juhl, L., Larsen, K. G., & Srba, J. (2011). Energy games in multiweighted automata. In A. Cerone & P. Pihlajasaari (Eds.), *Theoretical aspects of computing – ICTAC 2011* (pp. 95–115). Springer. https://doi.org/10.1007/978-3-642-23283-1_9
- Fahrenberg, U., & Legay, A. (2014). The quantitative linear-time–branching-time spectrum. *Theoretical Computer Science*, 538, 54–69. <https://doi.org/10.1016/j.tcs.2013.07.030>
- Fokkink, W., van Glabbeek, R., & Luttik, B. (2019). Divide and congruence III: From decomposition of modal formulas to preservation of stability and divergence. *Information and Computation*, 268, 104435. <https://doi.org/10.1016/j.ic.2019.104435>
- Ford, C., Milius, S., & Schröder, L. (2021). Behavioural preorders via graded monads. *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 1–13. <https://doi.org/10.1109/LICS52264.2021.9470517>
- Forster, J., Schröder, L., Wild, P., Beohar, H., Gurke, S., & Messing, K. (2024). Graded semantics and graded logics for eilenberg-moore coalgebras. In B. König & H. Urbat (Eds.), *Coalgebraic methods in computer science* (pp. 114–134). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-66438-0_6
- Fournet, C., & Gonthier, G. (2005). A hierarchy of equivalences for asynchronous calculi. *The Journal of Logic and Algebraic Programming*, 63(1), 131–173. <https://doi.org/j.jlap.2004.01.006>
- Frutos Escrig, D. de, Gregorio Rodríguez, C., Palomino, M., & Romero Hernández, D. (2013). Unifying the Linear Time–Branching Time Spectrum of Strong Process Semantics. *Logical Methods in Computer Science*, 9(2:11), 1–74. [https://doi.org/10.2168/LMCS-9\(2:11\)2013](https://doi.org/10.2168/LMCS-9(2:11)2013)
- Frutos Escrig, D. de, Keiren, J. J. A., & Willemse, T. A. C. (2017). Games for bisimulations and abstraction. *Logical Methods in Computer Science*, 13(4:15), 1–40. [https://doi.org/10.23638/LMCS-13\(4:15\)2017](https://doi.org/10.23638/LMCS-13(4:15)2017)

- Gale, D., & Stewart, F. M. (1953). Infinite games with perfect information. In H. W. Kuhn & A. W. Tucker (Eds.), *Contributions to the theory of games, volume II* (pp. 245–266). Princeton University Press. <https://doi.org/10.1515/9781400881970-014>
- Garavel, H. (2017). *The VLTS benchmark suite*. <https://doi.org/10.18709/perscido.2017.11.ds100>
- Garavel, H. (2023). *What is wrong with process calculi – and how to recover?* [Talk]. Open problems in concurrency theory (OPCT 2023), University of Urbino, Italy. <http://www.sti.uniurb.it/events/opct2023/SLIDES/garavel.pdf>
- Garavel, H., & Lang, F. (2022). Equivalence checking 40 years after: A review of bisimulation tools. In N. Jansen, M. Stoelinga, & P. van den Bos (Eds.), *LNCS: Vol. 13560. A journey from process algebra via timed automata to model learning: Essays dedicated to Frits Vaandrager on the occasion of his 60th birthday* (pp. 213–265). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-15629-8_13
- Garavel, H., Lang, F., Mateescu, R., & Serwe, W. (2013). CADP 2011: A toolbox for the construction and analysis of distributed processes. *International Journal on Software Tools for Technology Transfer*, 15(2), 89–107. <https://doi.org/10.1007/s10009-012-0244-z>
- Gazda, M., Fokkink, W., & Massaro, V. (2020). Congruence from the operator’s point of view: Syntactic requirements on modal characterizations. *Acta Informatica*, 57(3–5), 329–351. <https://doi.org/10.1007/s00236-019-00355-5>
- Geuvers, H. (2022). Apartness and distinguishing formulas in Hennessy–Milner logic. In N. Jansen, M. Stoelinga, & P. van den Bos (Eds.), *LNCS: Vol. 13560. A journey from process algebra via timed automata to model learning: Essays dedicated to Frits Vaandrager on the occasion of his 60th birthday* (pp. 266–282). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-15629-8_14
- Geuvers, H., & Golov, A. (2023). *Positive Hennessy-Milner logic for branching bisimulation*. <https://arxiv.org/abs/2210.07380>
- Geuvers, H., & Jacobs, B. (2021). Relating apartness and bisimulation. *Logical Methods in Computer Science*, 17(3). [https://doi.org/10.46298/LMCS-17\(3:15\)2021](https://doi.org/10.46298/LMCS-17(3:15)2021)
- Gibson-Robinson, T., Armstrong, P., Boulgakov, A., & Roscoe, A. W. (2014). FDR3: A modern refinement checker for CSP. In E. Ábrahám & K. Havelund (Eds.), *International conference on tools and algorithms for the construction and analysis of systems* (pp. 187–201). Springer. https://doi.org/10.1007/978-3-642-54862-8_13
- Göthel, T. (2012). *Mechanical verification of parameterized real-time systems* [PhD thesis, Berlin Institute of Technology]. <https://doi.org/10.14279/depositonce-3248>
- Groote, J. F., Jansen, D. N., Keiren, J. J. A., & Wijs, A. J. (2017). An $O(m \log n)$ algorithm for computing stuttering equivalence and branching bisimula-

- tion. *ACM Transactions on Computational Logic (TOCL)*, 18(2), 13:1–13:34. <https://doi.org/10.1145/3060140>
- Groote, J. F., & Martens, J. (2024). *A quadratic lower bound for simulation*. <https://doi.org/10.48550/arXiv.2411.14067>
- Groote, J. F., Martens, J., & Vink, Erik. P. de. (2023). Lowerbounds for bisimulation by partition refinement. *Logical Methods in Computer Science*, Volume 19, Issue 2. [https://doi.org/10.46298/lmcs-19\(2:10\)2023](https://doi.org/10.46298/lmcs-19(2:10)2023)
- Groote, J. F., & Mousavi, M. R. (2014). *Modelling and analysis of communicating systems*. MIT Press. <https://doi.org/10.7551/mitpress/9946.001.0001>
- Hauschild, J. (2023). *Nonlinear counterfactuals in Isabelle/HOL* [Bachelor's thesis, Technische Universität Berlin]. <https://github.com/johanneshauschild/NonlinearCounterfactuals>
- Hennessy, M., & Milner, R. (1980). On observing nondeterminism and concurrency. In J. W. de Bakker & J. van Leeuwen (Eds.), *LNCS: Vol. 85. Automata, languages and programming* (pp. 299–309). Springer. https://doi.org/10.1007/3-540-10003-2_79
- Hoare, C. A. R. (1985). *Communicating sequential processes*. Prentice-Hall, Inc.
- Hoare, C. A. R. (2006). Why ever CSP? *Electronic Notes in Theoretical Computer Science*, 162, 209–215. <https://doi.org/10.1016/j.entcs.2006.01.031>
- Horne, R., Mauw, S., & Yurkov, S. (2023). When privacy fails, a formula describes an attack: A complete and compositional verification method for the applied π -calculus. *Theoretical Computer Science*, 959, 113842. <https://doi.org/10.1016/j.tcs.2023.113842>
- Hülsbusch, M., König, B., Küpper, S., & Stoltenow, L. (2020). Conditional Bisimilarity for Reactive Systems. In Z. M. Ariola (Ed.), *LIPICs: Vol. 167. 5th international conference on formal structures for computation and deduction (FSCD 2020)* (pp. 10:1–10:19). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. <https://doi.org/10.4230/LIPICs.FSCD.2020.10>
- Hüttel, H., & Shukla, S. (1996). *On the complexity of deciding behavioural equivalences and preorders. A survey* (BRICS RS-96-39H). University of Aarhus. <https://doi.org/10.7146/brics.v3i39.20021>
- Kanellakis, P. C., & Smolka, S. A. (1983). CCS expressions, finite state processes, and three problems of equivalence. *PODC '83*, 228–240. <https://doi.org/10.1145/800221.806724>
- Keiren, J. J. A., & Willemse, T. A. C. (2024). It's all a game. In V. Capretta, R. Krebbers, & F. Wiedijk (Eds.), *Logics and type systems in theory and practice: Essays dedicated to Herman Geuvers on the occasion of his 60th birthday* (pp. 150–167). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-61716-4_10
- König, B., Mika-Michalski, C., & Schröder, L. (2020). Explaining non-bisimilarity in a coalgebraic approach: Games and distinguishing formulas. In D. Petrişan & J. Rot (Eds.), *LNCS: Vol. 12094. Coalgebraic methods in computer science: CMCS* (pp. 133–154). Springer. https://doi.org/10.1007/978-3-030-57201-3_8
- Kruskal, J. B. (1972). The theory of well-quasi-ordering: A frequently discov-

- ered concept. *Journal of Combinatorial Theory, Series A*, 13(3), 297–305. [https://doi.org/10.1016/0097-3165\(72\)90063-5](https://doi.org/10.1016/0097-3165(72)90063-5)
- Kučera, A., & Esparza, J. (1999). A logical viewpoint on process-algebraic quotients. In J. Flum & M. Rodríguez-Artalejo (Eds.), *LNCS: Vol. 1683. Computer science logic: CSL* (pp. 499–514). Springer. https://doi.org/10.1007/3-540-48168-0_35
- Kupferman, O., & Shamash Halevy, N. (2022). Energy Games with Resource-Bounded Environments. In B. Klin, S. Lasota, & A. Muscholl (Eds.), *LIPICs: Vol. 243. 33rd international conference on concurrency theory (CONCUR 2022)* (pp. 19:1–19:23). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. <https://doi.org/10.4230/LIPICs.CONCUR.2022.19>
- Kurzan, T. (2024). *Implementierung eines Contrasimilarity-Checkers für mCRL2* [Bachelor's thesis]. Technische Universität Berlin.
- Lange, M., Lozes, E., & Vargas Guzmán, M. (2014). Model-checking process equivalences. *Theoretical Computer Science*, 560(3), 326–347. <https://doi.org/10.1016/j.tcs.2014.08.020>
- Lê, H. N. (2020). *Implementing coupled similarity as an automated checker for mCRL2* [Bachelor's thesis]. Technische Universität Berlin.
- Lemke, C. (2024). *A formal proof of decidability of multi-weighted declining energy games* [Master's thesis, Technische Universität Berlin]. https://github.com/crmrtz/galois-energy-games/blob/main/master-thesis/masters_thesis.pdf
- Lönne, B. (2023). *An educational computer game about counterfactual truth conditions* [Bachelor's thesis, Technische Universität Berlin]. <https://github.com/Brunololos/counterfactuals-demo>
- Martens, J., & Groote, J. F. (2023). Computing Minimal Distinguishing Hennessy-Milner Formulas is NP-Hard, but Variants are Tractable. In G. A. Pérez & J.-F. Raskin (Eds.), *LIPICs: Vol. 279. 34th international conference on concurrency theory (CONCUR 2023)* (pp. 32:1–32:17). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. <https://doi.org/10.4230/LIPICs.CONCUR.2023.32>
- Martens, J., & Groote, J. F. (2024). Minimal depth distinguishing formulas without until for branching bisimulation. In V. Capretta, R. Krebbers, & F. Wiedijk (Eds.), *Logics and type systems in theory and practice: Essays dedicated to Herman Geuvers on the occasion of his 60th birthday* (pp. 188–202). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-61716-4_12
- Mattes, K. P. P. (2024). *Measuring expressive power of HML formulas in Isabelle/HOL* [Bachelor's thesis, Technische Universität Berlin]. https://github.com/ekeln/BA_formula_prices/raw/main/output/outline.pdf
- Mayr, R. (2000). Process rewrite systems. *Information and Computation*, 156(1), 264–286. <https://doi.org/10.1006/inco.1999.2826>
- Milner, R. (1980). *LNCS: Vol. 92. A calculus of communicating systems*. Springer. <https://doi.org/10.1007/3-540-10235-3>
- Milner, R. (1989). *Communication and concurrency*. Prentice-Hall, Inc.

- Mohri, M. (2002). Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3), 321–350. <https://doi.org/10.25596/jalc-2002-321>
- Montanari, L. (2021). *Kontrasimulation als Spiel* [Bachelor's thesis, Technische Universität Berlin]. <https://github.com/luisamontanari/ContrasimGame>
- Nestmann, U., & Pierce, B. C. (2000). Decoding choice encodings. *Information and Computation*, 163(1), 1–59. <https://doi.org/10.1006/inco.2000.2868>
- Ozegowski, F. (2023). *Integration eines generischen Äquivalenzprüfers in CAAL* [Bachelor's thesis, Technische Universität Berlin]. <https://github.com/Fabian-O01/CAAL>
- Paige, R., & Tarjan, R. E. (1987). Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6), 973–989. <https://doi.org/10.1137/0216062>
- Parrow, J., & Sjödin, P. (1992). Multiway synchronization verified with coupled simulation. In W. R. Cleaveland (Ed.), *CONCUR '92: Third international conference on concurrency theory stony brook, NY, USA, august 24–27, 1992 proceedings* (pp. 518–533). Springer. <https://doi.org/10.1007/BFb0084813>
- Peacock, D. A. (2020). *Process equivalences as a video game* [Bachelor's thesis, Technische Universität Berlin]. https://drive.google.com/drive/folders/19YuOCXI_7CEB4p9fgr6kDKCTbFwx2VU9
- Peterson, G. L. (1981). Myths about the mutual exclusion problem. *Inf. Process. Lett.*, 12, 115–116. [https://doi.org/10.1016/0020-0190\(81\)90106-X](https://doi.org/10.1016/0020-0190(81)90106-X)
- Pohlmann, M. (2021). *Reducing strong reactive bisimilarity to strong bisimilarity* [Bachelor's thesis, Technische Universität Berlin]. <https://maxpohlmann.github.io/Reducing-Reactive-to-Strong-Bisimilarity/thesis.pdf>
- Ranzato, F., & Tapparo, F. (2010). An efficient simulation algorithm based on abstract interpretation. *Information and Computation*, 208(1), 1–22. <https://doi.org/10.1016/j.ic.2009.06.002>
- Reed, J. N., Roscoe, A. W., & Sinclair, J. E. (2007). Responsiveness and stable revivals. *Formal Aspects of Computing*, 19, 303–319. <https://doi.org/10.1007/s00165-007-0032-9>
- Reichert, J. M. (2020). *Visualising and model checking counterfactuals* [Bachelor's thesis]. Technische Universität Berlin.
- Roscoe, A. W. (2009). Revivals, stuckness and the hierarchy of CSP models. *Journal of Logic and Algebraic Programming*, 78(3), 163–190. <https://doi.org/10.1016/j.jlap.2008.10.002>
- Sandt, E. (2022). *A video game about reactive bisimilarity* [Bachelor's thesis, Technische Universität Berlin]. <https://github.com/eloinoel/ReactiveBisimilarityGame>
- Sangiorgi, D. (1996). A theory of bisimulation for the π -calculus. *Acta Informatica*, 33(1), 69–97. <https://doi.org/10.1007/s002360050036>
- Sangiorgi, D. (2012). *Introduction to bisimulation and coinduction*. Cambridge University Press. <https://doi.org/10.1017/CBO9780511777110>
- Shukla, S. K., Hunt III, H. B., & Rosenkrantz, D. J. (1996). HORNSAT, model

- checking, verification and games: Extended abstract. In R. Alur & T. A. Henzinger (Eds.), *LNCS: Vol. 1102. Computer aided verification: CAV* (pp. 99–110). Springer. https://doi.org/10.1007/3-540-61474-5_61
- Spork, T., Baier, C., Katoen, J.-P., Piribauer, J., & Quatmann, T. (2024). A Spectrum of Approximate Probabilistic Bisimulations. In R. Majumdar & A. Silva (Eds.), *LIPICs: Vol. 311. 35th international conference on concurrency theory (CONCUR 2024)* (pp. 37:1–37:19). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. <https://doi.org/10.4230/LIPICs.CONCUR.2024.37>
- Steffen, B. (1989). Characteristic formulae. In G. Ausiello, M. Dezani-Ciancaglini, & S. R. Della Rocca (Eds.), *Automata, languages and programming* (pp. 723–732). Springer Berlin Heidelberg. <https://doi.org/10.1007/BFb0035794>
- Stirling, C. (1996). Modal and temporal logics for processes. In F. Moller & G. Birtwistle (Eds.), *LNCS: Vol. 1043. Logics for concurrency: Structure versus automata* (pp. 149–237). Springer. https://doi.org/10.1007/3-540-60915-6_5
- Stöcker, V. (2024). *Higher-order diadic μ -calculus—an efficient framework for checking process equivalences?* [Bachelor's thesis]. Technische Universität Berlin.
- Straßnick, T., & Ozegowski, F. (2024). *Integration of the weak spectroscopy into CAAL*. Technische Universität Berlin. https://github.com/equivio/CAAL/blob/spectroscopy/docs/CAAL_weak_Spectroscopy.pdf
- Tan, L. (2002a). An abstract schema for equivalence-checking games. In A. Cortesi (Ed.), *LNCS: Vol. 2294. Verification, model checking, and abstract interpretation, third international workshop, VMCAI 2002, venice, italy, january 21-22, 2002, revised papers* (pp. 65–78). Springer. https://doi.org/10.1007/3-540-47813-2_5
- Tan, L. (2002b). *Evidence-based verification*. State University of New York at Stony Brook. https://www.researchgate.net/publication/277283542_Evidence-Based_Verification
- Trzeciakiewicz, M. (2021). *Linear-time–branching-time spectroscopy as an educational web browser game* [Bachelor's thesis, Technische Universität Berlin]. <https://github.com/Marii19/the-spectroscopy-invaders>
- Valmari, A. (2009). Bisimilarity minimization in $O(m \log n)$ time. In G. Franceschinis & K. Wolf (Eds.), *Applications and theory of petri nets* (pp. 123–142). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-02424-5_9
- Valmari, A. (2020). All congruences below stability-preserving fair testing or CFFD. *Acta Informatica*, 57(3), 353–383. <https://doi.org/10.1007/s00236-019-00364-4>
- van Glabbeek, R. J. (1990). The linear time–branching time spectrum: Extended abstract. In J. C. M. Baeten & J. W. Klop (Eds.), *LNCS: Vol. 458. CONCUR'90* (pp. 278–297). Springer. <https://doi.org/10.1007/BFb0039066>
- van Glabbeek, R. J. (1993). The linear time–branching time spectrum II: The semantics of sequential systems with silent moves; extended abstract. In E. Best (Ed.), *LNCS: Vol. 715. CONCUR'93* (pp. 66–81). Springer. <https://doi.org/10.1007/BFb0039066>

- [//doi.org/10.1007/3-540-57208-2_6](https://doi.org/10.1007/3-540-57208-2_6)
- van Glabbeek, R. J. (2001). The linear time–branching time spectrum I: The semantics of concrete, sequential processes. In J. A. Bergstra, A. Ponse, & S. A. Smolka (Eds.), *Handbook of process algebra* (pp. 3–99). Elsevier. <https://doi.org/10.1016/B978-044482830-9/50019-9>
- van Glabbeek, R. J. (2023). Modelling mutual exclusion in a process algebra with time-outs. *Information and Computation*, 294, 105079. <https://doi.org/10.1016/j.ic.2023.105079>
- van Glabbeek, R. J., & Weijland, W. P. (1996). Branching time and abstraction in bisimulation semantics. *Journal of the ACM (JACM)*, 43(3), 555–600. <https://doi.org/10.1145/233551.233556>
- Vogel, G. (2024). *Accelerating process equivalence energy games using WebGPU* [Bachelor's thesis, Technische Universität Berlin]. <https://github.com/Gobbel2000/gpuequiv>
- Voorhoeve, M., & Mauw, S. (2001). Impossible futures and determinism. *Information Processing Letters*, 80(1), 51–58. [https://doi.org/10.1016/S0020-0190\(01\)00217-4](https://doi.org/10.1016/S0020-0190(01)00217-4)
- Wenzel, M. (2025). *The Isabelle/Isar reference manual*. <https://isabelle.in.tum.de/dist/Isabelle2025/doc/isar-ref.pdf>
- Wimmer, R. (2011). *Symbolische Methoden für die probabilistische Verifikation – Zustandsraumreduktion und Gegenbeispiele* [Dissertation, Albert-Ludwigs-Universität Freiburg]. <https://freidok.uni-freiburg.de/data/7932>
- Wimmer, R., Herbstritt, M., Hermanns, H., Strampp, K., & Becker, B. (2006). Sigref: A symbolic bisimulation tool box. In S. Graf & W. Zhang (Eds.), *Automated technology for verification and analysis: 4th international symposium, ATVA 2006, beijing, china, october 23-26, 2006. proceedings* (pp. 477–492). Springer. https://doi.org/10.1007/11901914_35
- Wißmann, T., Milius, S., & Schröder, L. (2021). Explaining Behavioural Inequivalence Generically in Quasilinear Time. In S. Haddad & D. Varacca (Eds.), *LIPICs: Vol. 203. 32nd international conference on concurrency theory (CONCUR 2021)* (pp. 32:1–32:18). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. <https://doi.org/10.4230/LIPICs.CONCUR.2021.32>
- Wittig, T. (2020). *Charting the jungle of process calculi encodings* [Bachelor's thesis, Technische Universität Berlin]. <https://concurrency-theory.org/process-jungle/>
- Wortmann, J. K., Olesen, S. R., & Enevoldsen, S. (2015). *CAAL 2.0 – recursive HML, distinguishing formulae, equivalence collapses and parallel fixed-point computations* (Vol. DES103F15) [Student project supervised by Jiří Srba and Kim Guldstrand Larsen]. Aalborg University. https://caal.cs.aau.dk/docs/CAAL2_RDEP.pdf
- Wrusch, M. (2020). *Ein Computerspiel zum Erlernen von Verhaltensäquivalenzen* [Bachelor's thesis]. Technische Universität Berlin.
- Zielonka, W. (1998). Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1), 135–183. [https://doi.org/10.1016/S0304-3975\(98\)00009-7](https://doi.org/10.1016/S0304-3975(98)00009-7)

Notation

Basics

iff	“if and only if”
$ A $	Number of elements of A
id_A	Identity relation on A
2^A	Power set of A
A^*, A^ω, A^∞	Language of finite / infinite / finite-or-infinite words over alphabet A
\mathcal{R}^{-1}	Inverse relation of \mathcal{R}
$\text{dom}(f)$	Domain of where function f is defined, \perp counts as undefined (first appearing in Definition 4.1)
$\mathbb{N}, \mathbb{N}_\infty$	Natural numbers / natural numbers extended by ∞ as top element
$:=$	Definitional equality; in the context of CCS, $X := P$ also means $\mathcal{V}(X) := P$ (Definition 2.3); in the context of algorithms, denotes variable assignment

Orders and Vectors

\leq	less-than-or-equal-to, usually pointwise on vectors
$\text{Min}(A), \text{Max}(A)$	Set of minimal / maximal elements in A in the context of a preorder \leq
$\sqcup / \sup, \sqcap / \inf$	Least upper bound / greatest lower bound in the context of a lattice (Definition 3.4)
$\uparrow A, \downarrow A$	Upward-closure / downward-closure of set A in the context of a preorder \leq (Definition 3.4)
$\hat{\mathbf{e}}_k$	Unit vector; component k is 1, all others 0 (in the context of some dimensionality d)
$\mathbf{0}$	Vector of zeros (in the context of some dimensionality d); not to be confused with $\mathbf{0}$ as terminated CCS process in Definition 2.2

Transition Systems

$\xrightarrow{\alpha}$	Transition relation (in context of a transition system, Definition 2.1), usually due to operational CCS semantics (Definition 2.3); can also connect <i>sets</i> of states
\nrightarrow^{α}	Absence of an α -transition
\rightarrow	Internal transition (Definition 6.1)
$\xrightarrow{(\alpha)}$	Soft transition (Definition 6.1)
$\xrightarrow{\vec{w}}$	Weak \vec{w} -word step sequence (Definition 6.1)
$\text{Der}(p, \alpha)$	α -derivatives of p (Definition 2.1)
$\text{Ini}(p)$	Enabled actions of p (Definition 2.1)
$\langle \alpha \rangle$	Modality of observing an α -step (Definition 2.11)
(α)	Modality of observing a τ -soft α -step (Definition 6.1 and 6.5)
$\langle \varepsilon \rangle$	Modality of allowing internal behavior (Definition 6.1 and 6.5)
$\llbracket \varphi \rrbracket$	Set of states that fulfill φ (Definition 2.12)
S/\sim	Quotient system on equivalence classes $[\cdot]_{\sim}$ (Definition 2.10)

Behavioral Equivalences

\preceq_N, \sim_N	Behavioral preorder / equivalence with respect to notion N (full names in Figure 3.8 and 6.5)
\sim_T, \sim_S, \sim_B	Strong trace equivalence, similarity, bisimilarity (Definition 2.5 and 2.7)
$\preceq_{\mathcal{O}}, \sim_{\mathcal{O}}$	Behavioral preorder / equivalence derived from a sublogic $\mathcal{O} \subseteq \text{HML}$ (Definition 2.13)
\mathcal{O}_N	Observation logic for a notion $N \in \mathbf{N}$ in the context of a spectrum (Definition 3.5)
$\mathcal{O}_{[B]}$	Bisimulation observation logic (Definition 2.16)
$\text{expr}(\varphi)$	Syntactic expressiveness price of φ in the context of a spectrum (Definition 3.6)
“ φ distinguishes”	Formula φ is true for state on left-hand side and not true for state(s) on right-hand side (Definition 2.13 and 7.3)

Games

\rightarrow	Game move (Definition 2.17)
\xrightarrow{u}	Game move, labeled by an energy update (Definition 4.1)
\mathcal{G}_B	Bisimulation game in the context of a transition system (Definition 2.22)
\mathcal{G}_B^\bullet	Bisimulation energy game (Definition 4.8)
$\text{Win}_a, \text{Win}_d$	Attacker / defender winning regions in the context of a reachability game (Definition 2.20), or winning budgets in an energy game (Definition 4.3)
$\text{Win}_a^{\min}(g)$	Minimal attacker winning budgets for position g in an energy game (Definition 4.3), solution to Problem 2
$\text{upd}, \text{upd}^\zeta$	Updates on energy games (Definition 4.1) and their undo-function in the algorithm for Galois energy games (Algorithm 4.1)
En	Energy levels on declining energy games in the context of a dimensionality d (Definition 4.5)
$\text{upd}, \text{upd}^\zeta$	Updates on declining energy games (Definition 4.5) and their undo-version (Definition 4.12)
\min_A	A special kind of update component in declining energy games (Definition 4.5)

Spectroscopy

$N_{p,q}$	Notions from N in the context of a spectrum preordering states p and q (solution to Problem 1)
\mathcal{G}_Δ	Strong spectroscopy game, in the context of a transition system (Definition 5.1)
$\mathcal{G}_\blacktriangle$	Clever spectroscopy game (Definition 5.4)
$\mathcal{G}_{\Delta N}, \mathcal{G}_{\blacktriangle N}$	Equivalence game for N , derived from a spectroscopy game (Definition 5.5)
\mathcal{G}_∇	Weak spectroscopy game, in the context of a transition system with silent steps (Definition 7.1)
$\mathcal{G}_{\widehat{\nabla}}$	Simplified weak spectroscopy game (Figure 7.3)
$\text{Strat}_B, \text{Strat}_\Delta, \text{Strat}_\nabla$	Strategy formulas in the context of polynomial / strong / weak spectroscopy (Definition 4.9, 5.2, 7.2)

Complexity

$O(f)$	Set of functions that grow to infinity at most as quickly as function f
P, NP, PSPACE	Complexity classes of deterministic polynomial time / nondeterministic polynomial time / polynomial space complexity

Recurring Examples

P, Q	Two systems of philosophers, differing in their determinism (Example 2.1, 2.2, 2.3)
T	Trolled philosophers with additional ☹-deadlock (Example 2.7)
Pe, Mx	Implementation of Peterson's mutual exclusion protocol and its specification (Example 1.1), styled Pe and Mx for their expression in the tool (Section 8.1.2)
