Generalized Equivalence Checking of Concurrent Programs

Benjamin Bisping
https://bbisping.de

Technische Universität Berlin

2025

Table of contents

Read this Book! (Once It’s Finished)

⚠ Warning

This is a work-in-progress document. Handle with care!

You should read this book (once it’s finished) if you’re interested in the semantics of concurrent programs and want to understand …

- How behavioral process equivalences are treated uniformly in equivalence spectra.
- How modal-logical and relational characterizations of behavioral equivalences are actually just “shadows” of game characterizations.
- How energy games can be used to turn a family of qualitative equivalence problems into one quantitative problems.
- How all behavioral equivalences can be decided at once.

TODO

(This is a space for TODOs that do not yet have a position in the text body.)

1 Introduction: What's the Difference?

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

I have run into this problem often, for instance when analyzing models of distributed algorithms or when devising examples for teaching. But obviously, the question already occurs every time one rewrites a program part and hopes for it to still do its job.

The first time I had to formally face the question was when working as a student research assistant: I was tasked with 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 the seminal paper on the landscape of process equivalences with internal behavior, the "Linear-time–branching-time spectrum II" by van Glabbeek (1993). Its central figure, reproduced in Figure 1.1, mesmerized me. But, how to use the spectrum to decide for a given pair of processes which of the many equivalences apply, is far from straightforward. Over the years, I have learned that others too have run into this problem: For instance, Nestmann & Pierce (2000) thinking about process algebra encodings and Bell (2013) verifying compiler optimizations. The problem can be summarized as follows:

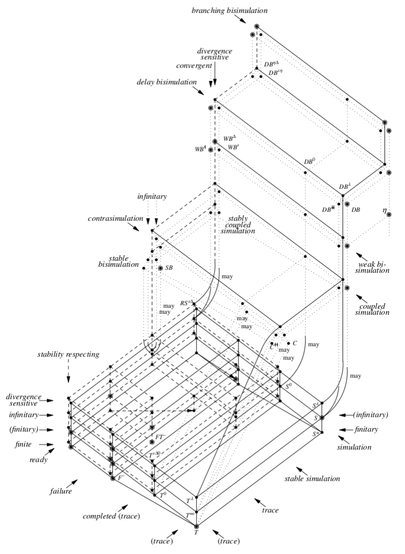


Figure 1.1: The linear-time–branching-time spectrum with silent moves as depicted in Glabbeek (1993).

[1] The aim was to bridge between the tools FDR2 and UPPAAL for Göthel's dissertation (2012). 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!

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

Above question will be the *research question of this thesis*. We[2] 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.

[2] As you might have noticed, this text uses more personal pronouns than is common in much of computer science literature. Their meaning is the same as in a lecture: "I" = "The author"; "You" = "The audience"; "We" = "The asynchronous collective of author and readers."

1.1 Linear-Time–Branching-Time Spectroscopy

Van Glabbeek's papers on comparative concurrency semantics (1990, 1993) 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. He unveils an underlying structure where the questions can easily be compared with respect to their distinctive power. This is analogous to the *spectrum of light* where seemingly qualitative properties ("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 may ask:

> If there exists a "linear-time–branching time spectrum," does this mean, there also is some kind of "linear-time–branching-time *spectroscopy*"?

This thesis answers the 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 processes, and this places the two on the spectrum of equivalences. We thus turn a set of qualitative equivalence problems into one quantitative problem of *how equivalent* two systems are. As we will see, this amounts to an abstract form of subtraction between programs, determining what kinds of differences an outside examiner might observe.

1.2 This Thesis

The algorithm at the core of this thesis describes how to *decide all behavioral equivalences at once* for varying spectra of equivalence using *games that can count* to *limit possible distinctions by attacker energy levels*. More precisely, we make the following contributions:

- Chapter 2 lays some foundations and makes precise *how bisimilarity games relate to grammars of distinguishing formulas* of Hennessy–Milner modal logic.
- Chapter 3 shows *how to understand the strong linear-time–branching-time spectrum quantitatively* and formalizes the *spectroscopy problem*.
- Chapter 4 introduces the approach of *characterizing equivalence spectra through a class of energy games* and *how to decide such games*.
- TODO Chapter 5 applies the approach to decide the whole strong spectrum through one *game for linear-time–branching-time spectroscopy*.
- TODO Chapter 6 adapt the game for the *weak spectrum of equivalences accounting for silent-steps*.
- TODO Chapter 7 showcases implementations to conveniently perform equivalence spectroscopies in web browsers and using GPU parallelization.

1.3 Artifacts and Papers

This thesis ties together the work of several publications in a coherent presentation. It is written to be understandable on its own. For details, we usually 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) 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 2023a) makes a *big technical leap by using energy games*, removing the necessity for explicit formula construction. (Tech report: arXiv 2023b)
- **"Characterizing Contrasimilarity through Games, Modal Logic, and Complexity"** (Information & Computation 2024) closes *gaps in the weak spectrum of equivalences* for games and their complexity and their link 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) adapts the *spectroscopy approach for the weak spectrum.*

Prototype. The algorithms of this thesis have been validated through a *Scala prototype implementation.* It can be tried out in the browser on https://equiv.io. The source is available openly on https://github.com/benkeks/equivalence-fiddle.

Isabelle formalization. Instead of swamping the document with technical proofs, these are contained in machine-checkable Isabelle/HOL theories.

- **Equivalence spectrum formalization** contains proofs for the early chapters: https://benkeks.github.io/ltbt-spectroscopy-isabelle/.
- **Weak Equivalence Spectroscopy** formalizes the core results of Chapter 6: https://equivio.github.io/silent-step-spectroscopy. The theory has been developed together with TU Berlin students Lisa Annett Barthel, Leonard Moritz Hübner, Caroline Lemke, Karl Parvis Philipp Mattes, and Lenard Mollenkopf.

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 Subsection 7.3.1.
- Ozegowski (2023): *Integration eines generischen Äquivalenzprüfers in CAAL* extends the Concurrency Workbench Aalborg Edition with a spectroscopy feature, reported in Subsection 7.3.2.

- 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 WebGPU*, topic of Section 7.2, allows massive parallelization on the GPU of key parts of our algorithm.
- Lemke (2024): *A Formal Proof of Decidability of Multi-Weighted Declining Energy Games* formalizes the fixedpoint algorithm of Section 4.3.

Other publications. Only indirectly connected to this dissertation project are my prior publications in leading me to topic and techniques:

- Bisping et al. (2016, ITP): "Mechanical Verification of a Constructive Proof for FLP."
- Bisping & Nestmann (2019, TACAS): "Computing Coupled Similarity."
- Bisping et al. (2020, Acta Informatica): "Coupled similarity: the first 32 years."

Other Bachelor theses. During my dissertation project, I also supervised several other Bachelor theses, many of which played important roles in shaping the research. Although they do not appear directly on the following pages, I want to acknowledge the students' vital contribution to the 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*

2 Preliminaries: Communicating Systems and Games

There is a virtually infinite supply of formalisms to model programs and communicating systems. They are accompanied by many notions for behavioral equivalence and refinement to relate or distinguish their models.

This chapter takes a tour into this field. The tour is *agnostic* in building on the most basic and versatile formalism of *labeled transition systems* and on standard equivalences such as *trace equivalence* and *bisimilarity*. The approach is also *opinionated* in focussing on Robin Milner's tradition of concurrency theory with a strong pull towards *game characterizations*. The very core concepts are described in Section 2.1.

We will explore the trinity of relational, modal, and game-theoretic characterizations of behavioral equivalences. Figure 2.1 shows the scaffold of this section, 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 *Hennessy–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 states apart and a defender tries to prevent this.

Readers who are familiar with the contents of Figure 2.1 can mostly skim through this chapter of preliminaries. The two core insights that this chapter intends to seed for coming chapters are:

ℹ Idea 1: It's all a game!

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

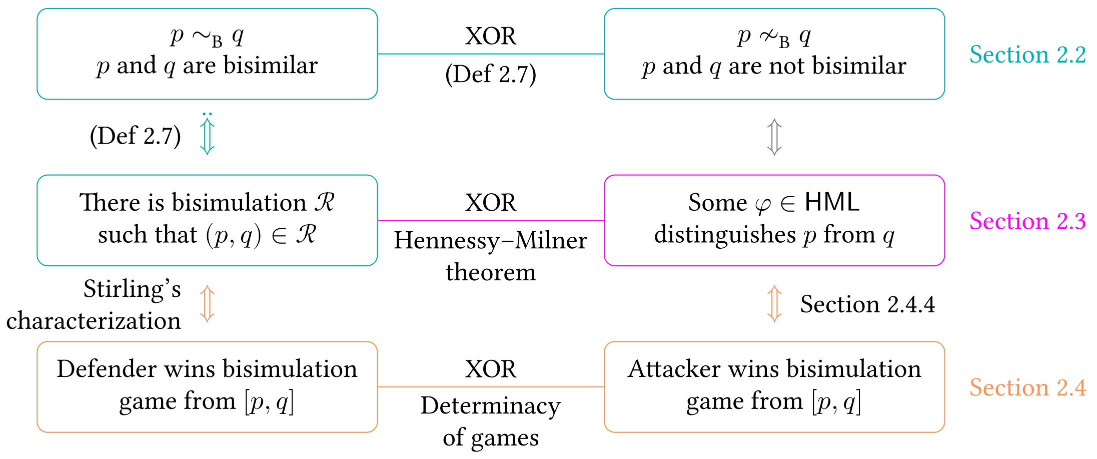

$p \sim_B q$
p and q are bisimilar
XOR
(Def 2.7)
$p \nsim_B q$
p and q are not bisimilar
Section 2.2
(Def 2.7)
There is bisimulation $\mathcal{R}$ such that $(p, q) \in \mathcal{R}$
XOR
Hennessy–Milner theorem
Some $\varphi \in$ HML distinguishes p from q
Section 2.3
Stirling's characterization
Section 2.4.4
Defender wins bisimulation game from $[p, q]$
XOR
Determinacy of games
Attacker wins bisimulation game from $[p, q]$
Section 2.4

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

Idea 2: 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 who are more used 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 the bridge, we will not go back.

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,[3] 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.

[3] But some are useful, as the saying goes.

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 where the finiteness constraint has been lifted.

Definition 2.1 (Transition Systems). A *labeled transition system* (LTS) $\mathcal{S} = (\mathcal{P}, \mathit{Act}, \rightarrow)$ consists of

- $\mathcal{P}$, a set of *states*,

- *Act*, a set of *actions*, and
- $\rightarrow \subseteq \mathcal{P} \times \mathit{Act} \times \mathcal{P}$, a *transition relation*.[4]

[4] locale Labeled_Transition_Systems.lts

We write $\mathrm{Der}(p, \alpha)$ for the set of derivative states $\{p' \mid p \xrightarrow{\alpha} p'\}$, and $\mathrm{Ini}(p)$ for the set of enabled actions $\{\alpha \mid \exists p'.\, p \xrightarrow{\alpha} p'\}$.

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.

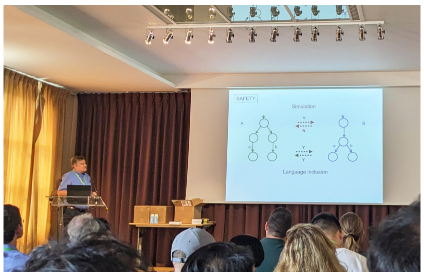


Figure 2.2: Tom Henzinger employing Example 2.1 during CAV'23.

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

$$\begin{tikzcd}
& \mathsf{P} \arrow[dl, "\tau"'] \arrow[dr, "\tau"] & & & & \mathsf{Q} \arrow[d, "\tau"] & \\
\mathsf{p_a} \arrow[d, "\mathsf{a}"] & & \mathsf{p_b} \arrow[d, "\mathsf{b}"] & & & \mathsf{q_{ab}} \arrow[dl, "\mathsf{a}"'] \arrow[dr, "\mathsf{b}"] & \\
\mathsf{p_1} & & \mathsf{p_2} & & \mathsf{q_1} & & \mathsf{q_2}
\end{tikzcd}$$

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

The program described by the transitions from P choses non-deterministically during a τ-step between two options and then offers only either a *or* b. The program 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 non-determinism in programs? Subsection 2.1.2 shows how non-determinism arises naturally in concurrent programs.
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 in later chapters stand for *internal behavior* and receive special treatment. For the scope of this and the following three chapters, τ is an action like every other.

Generally, this thesis aims to be consistent with notation and naming in surrounding literature. Where there are different options, we usually prefer the less-greek one. Also, we will usually write literals and constant names in sans-serif and variables in *italics*. For the internal action, the whole field has converged to τ in italics, however—so, we will run with this.

2.1.2 Calculus of Communicating Systems

To talk about programs in this thesis, we use Milner's (1989) *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!

Definition 2.2 (Calculus of Communicating Systems). Let $\mathcal{A}$ be a set of channel names, and $\mathcal{X}$ a set of process names. Then, CCS processes, communicating via actions $\mathsf{Act}_{\mathsf{CCS}} := \mathcal{A} \cup \{\overline{\alpha} \mid \alpha \in \mathcal{A}\} \cup \{\tau\}$, are given by the following grammar:

$$\begin{array}{rlll} P ::= & \alpha.P & \text{with } \alpha \in \mathcal{A} & \text{“input action prefix”} \\ & \overline{\alpha}.P & \text{with } \alpha \in \mathcal{A} & \text{“output action prefix”} \\ & \tau.P & & \text{“internal action”} \\ & \mathbf{0} & & \text{“null process”} \\ & X & \text{with } X \in \mathcal{X} & \text{“recursion”} \\ & P + P & & \text{“choice”} \\ & P \mid P & & \text{“parallel composition”} \\ & P \setminus A & \text{with } A \subseteq \mathcal{A} & \text{“restriction”} \end{array}$$

We call pairs of actions α and $\overline{\alpha}$ *coactions*, and work with the aliasing $\overline{\overline{\alpha}} = \alpha$. The intuition is that an action α represents receiving and $\overline{\alpha}$ expresses sending in communication. A pair of action and coaction can “react” in a communication situation and only become internal activity τ in the view of the environment.

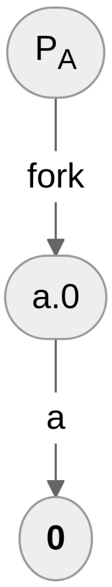
Each sub-process tree must end in a $\mathbf{0}$-process or recursion. For brevity, we usually drop a final $\mathbf{0}$ when writing terms, e.g., just writing ac for $\mathsf{ac}.\mathbf{0}$.

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.

Example 2.2 (Concurrent Philosophers). Following tradition, we will express our examples in terms of philosophers who need forks to eat spaghetti.[5] So, consider two philosophers $\mathsf{P_A}$ and $\mathsf{P_B}$ who want to grab a resource fork modeled as an action in order to eat where we express $\mathsf{P_A}$ eating with action a and $\mathsf{P_B}$ eating with b. The philosopher processes read:

[5] Of course, you can just as well read the examples to be about computer programs that race for resources.

$$\mathsf{P_A} := \mathsf{fork}.\mathsf{a}.\mathbf{0}$$
$$\mathsf{P_B} := \mathsf{fork}.\mathsf{b}.\mathbf{0}$$



An LTS representation of $\mathsf{P_A}$'s behavior can be seen in the margin. Process P captures the whole scenario where the two philosophers compete for the fork using communication:

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

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

As the $\overline{\mathsf{fork}}$-action 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.

The formal relationship between process terms and their LTS semantics is given by the following definition.

Definition 2.3 (CCS Semantics). Given an assignment of names to processes, $\mathcal{V} \colon \mathcal{X} \to \mathsf{CCS}$, the operational semantics $\xrightarrow{\cdot}_{\mathsf{CCS}} \subseteq \mathsf{CCS} \times \mathsf{Act}_{\mathsf{CCS}} \times \mathsf{CCS}$ is defined inductively by the rules:

$$\frac{}{\alpha.P \xrightarrow{\alpha}_{\mathsf{CCS}} P} \qquad \frac{P \xrightarrow{\alpha}_{\mathsf{CCS}} P' \qquad \mathcal{V}(X) = P}{X \xrightarrow{\alpha}_{\mathsf{CCS}} P'}$$

$$\frac{P_1 \xrightarrow{\alpha}_{\mathsf{CCS}} P_1'}{P_1 + P_2 \xrightarrow{\alpha}_{\mathsf{CCS}} P_1'} \qquad \frac{P_2 \xrightarrow{\alpha}_{\mathsf{CCS}} P_2'}{P_1 + P_2 \xrightarrow{\alpha}_{\mathsf{CCS}} P_2'}$$

$$\frac{P_1 \xrightarrow{\alpha}_{\mathsf{CCS}} P_1'}{P_1 \mid P_2 \xrightarrow{\alpha}_{\mathsf{CCS}} P_1' \mid P_2} \qquad \frac{P_2 \xrightarrow{\alpha}_{\mathsf{CCS}} P_2'}{P_1 \mid P_2 \xrightarrow{\alpha}_{\mathsf{CCS}} P_1 \mid P_2'}$$

$$\frac{P_1 \xrightarrow{\alpha}_{\mathsf{CCS}} P_1' \qquad P_2 \xrightarrow{\overline{\alpha}}_{\mathsf{CCS}} P_2'}{P_1 \mid P_2 \xrightarrow{\tau}_{\mathsf{CCS}} P_1' \mid P_2'} \qquad \frac{P \xrightarrow{\alpha}_{\mathsf{CCS}} P' \qquad \alpha, \overline{\alpha} \notin A}{P \setminus A \xrightarrow{\alpha}_{\mathsf{CCS}} P' \setminus A}$$

A process $P \in \mathsf{CCS}$ now denotes a position in the transition system $(\mathsf{CCS}, \mathsf{Act}_{\mathsf{CCS}}, \rightarrow_{\mathsf{CCS}})$ defined through Definition 2.2.

Feel free to go ahead an 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 transition $\mathsf{P} \xrightarrow{\tau} \mathsf{p_a}$ of Example 2.1 would be justified as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{\mathsf{fork}} \xrightarrow{\overline{\mathsf{fork}}}_{\mathsf{CCS}} \mathbf{0}} \qquad \dfrac{\dfrac{\dfrac{}{\mathsf{fork.a} \xrightarrow{\mathsf{fork}}_{\mathsf{CCS}} \mathsf{a}} \qquad \mathcal{V}(\mathsf{P_A}) = \mathsf{fork.a}}{\mathsf{P_A} \xrightarrow{\mathsf{fork}}_{\mathsf{CCS}} \mathsf{a}}}{\mathsf{P_A} \mid \mathsf{P_B} \xrightarrow{\mathsf{fork}}_{\mathsf{CCS}} \mathsf{a} \mid \mathsf{P_B}}}{\overline{\mathsf{fork}} \mid \mathsf{P_A} \mid \mathsf{P_B} \xrightarrow{\tau}_{\mathsf{CCS}} \mathbf{0} \mid \mathsf{a} \mid \mathsf{P_B}} \qquad \tau \notin \{\mathsf{fork}\}}{(\overline{\mathsf{fork}} \mid \mathsf{P_A} \mid \mathsf{P_B}) \setminus \{\mathsf{fork}\} \xrightarrow{\tau}_{\mathsf{CCS}} (\mathbf{0} \mid \mathsf{a} \mid \mathsf{P_B}) \setminus \{\mathsf{fork}\}} \qquad \mathcal{V}(\mathsf{P}) = (\overline{\mathsf{fork}} \mid \mathsf{P_A} \mid \mathsf{P_B}) \setminus \{\mathsf{fork}\}}{\mathsf{P} \xrightarrow{\tau}_{\mathsf{CCS}} (\mathbf{0} \mid \mathsf{a} \mid \mathsf{P_B}) \setminus \{\mathsf{fork}\}}$$

Non-determinism like 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 non-determinism plays a crucial role in models, for instance, as consequence of abstraction or 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:

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

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*. Clearly, there might be 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.

This section will quickly introduce 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/ouput events they might expose, referred to as their *traces*, is a natural starting point to talk about behavioral equivalences.

Definition 2.4 (Traces). The set of traces of a state $\mathsf{Traces}(p)$ is recursively defined as

- $() \in \mathsf{Traces}(p)$,[6]
- $\alpha \cdot \vec{w} \in \mathsf{Traces}(p)$ if there is p' with $p \xrightarrow{\alpha} p'$ and $\vec{w} \in \mathsf{Traces}(p')$.[7]

Definition 2.5 (Trace Equivalence). Two states p and q are considered *trace-equivalent*, written $p \sim_\mathrm{T} q$, if $\mathsf{Traces}(p) = \mathsf{Traces}(q)$.[8]

States are *trace-preordered*, $p \preceq_\mathrm{T} q$, if $\mathsf{Traces}(p) \subseteq \mathsf{Traces}(q)$.[9]

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

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

[6] We denote the empty word by $()$.

[7] abbreviation Labeled_Transition_Systems.lts.traces

[8] abbreviation Strong_Equivalences.lts.trace_equivalent

[9] definition Strong_Equivalences.lts.trace_preordered

Proposition 2.1. *The trace preorder $\preceq_{\mathrm{T}}$ is indeed a preorder (i.e., transitive and reflexive)*[10] *and trace equivalence $\sim_{\mathrm{T}}$ is indeed an equivalence relation (i.e., transitive, reflexive, and moreover symmetric).*[11]

[10] lemma Strong_Equivalences.lts.trace_preorder_transitive

[11] lemma Strong_Equivalences.lts.trace_equivalence_equiv

Trace equivalence and preorder assign programs straightforward *denotational semantics*: The sets of traces they might expose. For many 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 execute.

There are several reasons why computer scientist did content semselves 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. For instance, assume there is a user that wants to communicate $(\cdots \mid \mathsf{a}.\mathsf{success}.\mathbf{0}) \setminus \{\mathsf{a}\}$. In interactions with Q, they will always reach success; with P, there is a possibility of ending up deadlocked in parallel with $\mathsf{p_b}$, never achieving success.

2.2.2 Simulation and Bisimulation

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 another.

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 $\alpha \in \mathit{Act}$ with $p \xrightarrow{\alpha} p'$ there is a q' with $q \xrightarrow{\alpha} q'$ and $(p', q') \in \mathcal{R}$.[12]

[12] definition Strong_Equivalences.lts.simulation

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

- p is *simulated by* q, $p \preceq_{\mathrm{S}} q$, iff there is a simulation $\mathcal{R}$ with $(p, q) \in \mathcal{R}$.[13]
- p is *similar* to q, $p \sim_{\mathrm{S}} q$, iff $p \preceq_{\mathrm{S}} q$ and $q \preceq_{\mathrm{S}} p$.[14]
- p is *bisimilar* to q, $p \sim_{\mathrm{B}} q$, iff there is a *symmetric* simulation $\mathcal{R}$ (i.e. $\mathcal{R} = \mathcal{R}^{-1}$) with $(p, q) \in \mathcal{R}$.[15]

[13] definition Strong_Equivalences.lts.simulated_by

[14] abbreviation Strong_Equivalences.lts.similar

[15] definition Strong_Equivalences.lts.bisimilar

We also call a symmetric simulation *bisimulation* for short.[16] Canceled symbols of relations refer to their negations, for instance, $p \not\sim_{\mathrm{S}} q$ iff there is *no* simulation $\mathcal{R}$ with $(p, q) \in \mathcal{R}$.

[16] 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.

Example 2.5. The following relations are simulations on the LTS of Example 2.1:

- the empty relation $\mathcal{R}_{\emptyset} := \emptyset$;

- the identity relation $\mathcal{R}_{\text{id}} := \text{id}_{\{\mathsf{P},\mathsf{p_a},\mathsf{p_b},\mathsf{p_1},\mathsf{p_2},\mathsf{Q},\mathsf{q_{ab}},\mathsf{q_1},\mathsf{q_2}\}} = \{(\mathsf{P},\mathsf{P}),(\mathsf{p_a},\mathsf{p_a}),$ $(\mathsf{p_b},\mathsf{p_b}),(\mathsf{p_1},\mathsf{p_1}),(\mathsf{p_2},\mathsf{p_2}),(\mathsf{Q},\mathsf{Q}),(\mathsf{q_{ab}},\mathsf{q_{ab}}),(\mathsf{q_1},\mathsf{q_1}),(\mathsf{q_2},\mathsf{q_2})\}$;
- the universal relation between all final states $\mathcal{R}_{\text{fin}} := \{\mathsf{p_1},\mathsf{p_2},\mathsf{q_1},\mathsf{q_2}\}^2$,
- more generally, the relation from final states to all other states: $\mathcal{R}_{\text{up}} := \{\mathsf{p_1},\mathsf{p_2},\mathsf{q_1},\mathsf{q_2}\} \times \mathcal{P}$;
- a minimal simulation for P and Q: $\mathcal{R}_{\mathsf{PQ}} := \{(\mathsf{P},\mathsf{Q}),(\mathsf{p_a},\mathsf{q_{ab}}),(\mathsf{p_b},\mathsf{q_{ab}}),(\mathsf{p_1},\mathsf{q_1}),(\mathsf{p_2},\mathsf{q_2})\}$;
- and the combination of the above $\mathcal{R}_{\text{max}} := \mathcal{R}_{\text{sim}} \cup \mathcal{R}_{\text{id}} \cup \mathcal{R}_{\text{up}}$.

The simulation $\mathcal{R}_{\mathsf{PQ}}$ shows that $\mathsf{P} \preceq_{\mathrm{S}} \mathsf{Q}$.

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

Thus, $\mathsf{Q} \not\preceq_{\mathrm{S}} \mathsf{P}$, and $\mathsf{P} \not\sim_{\mathrm{S}} \mathsf{Q}$. Moreover, there cannot be a symmetric simulation, $\mathsf{P} \not\sim_{\mathrm{B}} \mathsf{Q}$.

Proposition 2.2. *The simulation preorder $\preceq_{\mathrm{S}}$ is indeed a preorder*[17]*, and $\sim_{\mathrm{S}}$ and $\sim_{\mathrm{B}}$ are equivalences.*[18]

[17] lemma Strong_Equivalences.lts.simulation_preorder_transitive

[18] lemma Strong_Equivalences.lts.bisimilarity_equiv

2.2.3 The Hierarchy of Equivalences

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 $\sim_{\mathrm{B}}$ is itself a symmetric simulation.*[19]

Corollary 2.1. *If $p \sim_{\mathrm{B}} q$, then $p \sim_{\mathrm{S}} q$.*[20]

Lemma 2.2. *If $p \preceq_{\mathrm{S}} q$, then $p \preceq_{\mathrm{T}} q$.*[21] *(Consequently, $p \sim_{\mathrm{S}} q$ also implies $p \sim_{\mathrm{T}} q$.*[22]*)*

We also have seen with example Example 2.5 that this hierarchy is strict between trace and simulation preorder in the sense that there exist p, q with $p \preceq_{\mathrm{T}} q$ but not $p \preceq_{\mathrm{S}} q$. The hierarchy also is strict between similarity and bisimilarity as the following example shows.

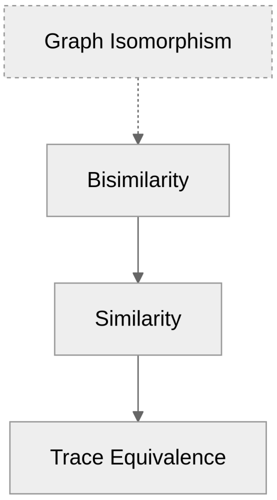


Figure 2.4: Core hierarchy of equivalences.

[19] lemma Strong_Equivalences.lts.bisim_bisim

[20] lemma Strong_Equivalences.lts.bisim_bisim

[21] lemma Strong_Equivalences.lts.sim_implies_trace_preord

[22] lemma Strong_Equivalences.lts.sim_eq_implies_trace_eq

Example 2.6 (Trolled philosophers). Let us extend Q of Example 2.3 to include a troll process that might consume the fork and then do nothing:

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

This adds another deadlock state to the transition system, seen in Figure 2.5.

To similarity, this change is invisible, that is $\mathsf{Q} \sim_{\mathrm{S}} \mathsf{T}$. (Reason: The relation $\{(\mathsf{Q},\mathsf{T}),(\mathsf{T},\mathsf{Q})\} \cup \text{id}_{\{\mathsf{q_{ab}},\mathsf{q_1},\mathsf{q_2},\mathsf{q_3}\}}$ is a simulation.)

However, to bisimilarity, $\mathsf{T} \xrightarrow{\tau} \mathsf{q_3}$ constitutes a difference. There cannot be a symmetric simulation handling this transition as Q has no deadlocked successors. Thus $\mathsf{Q} \not\sim_{\mathrm{B}} \mathsf{T}$.

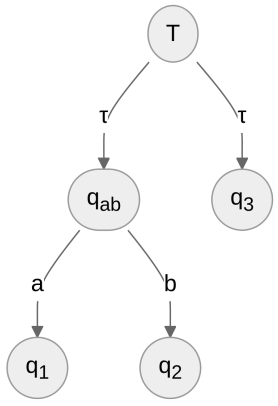


Figure 2.5: Example with new deadlock $\mathsf{q_3}$.

The equivalences we have been 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 $f\colon \mathcal{P} \to \mathcal{P}$ 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.[23]

Two states p and q are considered *graph-isomorphism-equivalent*, $p \sim_{\mathrm{ISO}} q$, iff there is a graph isomorphism f with $f(p) = q$.[24]

[23] definition Strong_Equivalences.lts.isomorphism

[24] definition Strong_Equivalences.lts.is_isomorphic_to

Example 2.7. Consider the transition system in Figure 2.6. $\mathsf{p}_{\mathsf{even}} \sim_{\mathrm{ISO}} \mathsf{p}_{\mathsf{odd}}$ because $f := \{\mathsf{p}_{\mathsf{even}} \mapsto \mathsf{p}_{\mathsf{odd}}, \mathsf{p}_{\mathsf{odd}} \mapsto \mathsf{p}_{\mathsf{even}}\}$ is a graph isomorphism.

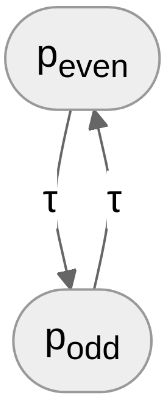


Figure 2.6: Transition system with an isomorphic cycle.

Lemma 2.3. *The relation $\sim_{\mathrm{ISO}}$ is itself a symmetric simulation and thus $p \sim_{\mathrm{ISO}} q$ implies $p \sim_{\mathrm{B}} q$.*[25]

[25] lemma Strong_Equivalences.lts.iso_implies_bisim

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

Example 2.8. Consider the processes $\mathsf{P}_1 := (\overline{\mathsf{fork}} \mid \mathsf{fork}) \setminus \{\mathsf{fork}\}$ and $\mathsf{P}_2 := (\overline{\mathsf{fork}} \mid \mathsf{fork} \mid \mathsf{fork}) \setminus \{\mathsf{fork}\}$. P_1 can transition to $(\mathbf{0} \mid \mathbf{0}) \setminus \{\mathsf{fork}\}$, while P_2 has two options, namely $(\mathbf{0} \mid \mathbf{0} \mid \mathsf{fork}) \setminus \{\mathsf{fork}\}$ and $(\mathbf{0} \mid \mathsf{fork} \mid \mathbf{0}) \setminus \{\mathsf{fork}\}$. All three reachable processes are deadlocks and thus isomorphic. But $\mathsf{P}_1 \not\sim_{\mathrm{ISO}} \mathsf{P}_2$ because no bijection can connect the one successor of P_1 and the two of P_2. However, $\mathsf{P}_1 \sim_{\mathrm{B}} \mathsf{P}_2$, as bisimilarity is more relaxed.

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 $f\colon B_1 \times \dots \times B_n \to C$ is called *monotonic* with respect to a family of partial orders $(B_k, \leq_k)$ 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 $b_k \leq b'_k$ for all $k \leq n$ implies that $f(b) \leq f(b')$. We will usually 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 2.9 (Parity as Congruence). As a standard example for a congruence, consider the equivalence relation of equally odd numbers

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

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

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

Lemma 2.4. *The operators of CCS (Definition 2.2) are congruences for trace equivalence and bisimilarity on the CCS transition system (Definition 2.3).*

Proof. This is a well-established fact. TODO: Citation!

As an example, let us prove the congruence property for choice, $+$, on bisimilarity, $\sim_{\mathrm{B}}$. For any CCS context, we assume $P \sim_{\mathrm{B}} R$ and $Q \sim_{\mathrm{B}} T$. We have to show that $P + Q \sim_{\mathrm{B}} R + T$, that is due to symmetry, that for any α with $P + Q \xrightarrow{\alpha}_{\mathsf{CCS}} PQ'$ there is RT' such that $R + T \xrightarrow{\alpha}_{\mathsf{CCS}} RT'$ and $PQ' \sim_{\mathrm{B}} RT'$. By the semantics of CCS, the first step must be either due to $P \xrightarrow{\alpha}_{\mathsf{CCS}} PQ'$ or, analogously, $Q \xrightarrow{\alpha}_{\mathsf{CCS}} PQ'$. Without loss of generality, we only consider $P \xrightarrow{\alpha}_{\mathsf{CCS}} PQ'$. Due to $P \sim_{\mathrm{B}} R$, this step implies some R' with $R \xrightarrow{\alpha}_{\mathsf{CCS}} R'$ and $PQ' \sim_{\mathrm{B}} R'$. As $R + T \xrightarrow{\alpha}_{\mathsf{CCS}} R'$, we can take this R' to prove our goal. □

2.3 Modal Logic

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."

We care about modal logics as they too can be used to characterize behavioral equivalences. In this section, we will show how this characterization works and argue that, for our purposes of comparative semantics, modal characterizations are a superb medium.

2.3.1 Hennessy–Milner Logic to Express Observations

Hennessy & Milner (1980) introduce the modal logic that is now commonly called *Hennessy–Milner logic* 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 2.10 (Hennessy–Milner Logic). Formulas of *Hennessy–Milner logic* HML are given by the grammar:[26]

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

[26] datatype Hennessy_Milner_Logic.hml _formula

We also write conjunctions as sets $\bigwedge\{\varphi_1, \varphi_2 ...\}$. 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 $\langle \mathsf{a}\rangle\langle \mathsf{b}\rangle\top$ to $\langle \mathsf{a}\rangle\langle \mathsf{b}\rangle$.

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

Definition 2.11 (HML semantics). The semantics of HML $[\![\cdot]\!] \colon \mathsf{HML} \to 2^{\mathcal{P}}$ is defined recursively by:[27]

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

Example 2.10. Let us consider some observations on the system of Example 2.1.

- $[\![\langle\tau\rangle\langle\mathsf{a}\rangle]\!] = \{\mathsf{P}, \mathsf{Q}\}$ as both, P and Q, expose the trace $\tau\mathsf{a}$,
- $\mathsf{Q} \in [\![\langle\tau\rangle \bigwedge\{\langle\mathsf{a}\rangle, \langle\mathsf{b}\rangle\}]\!]$, but $\mathsf{P} \notin [\![\langle\tau\rangle \bigwedge\{\langle\mathsf{a}\rangle, \langle\mathsf{b}\rangle\}]\!]$.
- $\mathsf{P} \in [\![\langle\tau\rangle\neg\langle\mathsf{a}\rangle]\!]$, but $\mathsf{Q} \notin [\![\langle\tau\rangle\neg\langle\mathsf{a}\rangle]\!]$.

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 2.12 (Distinctions and Equivalences). We say that formula $\varphi \in \mathsf{HML}$ *distinguishes* state p *from* state q if $p \in [\![\varphi]\!]$ but $q \notin [\![\varphi]\!]$.[28]

We say a sublogic $\mathcal{O} \subseteq \mathsf{HML}$ *preorders* state p to q, $p \preceq_{\mathcal{O}} q$, if no $\varphi \in \mathcal{O}$ is distinguishing p from q.[29] If the preordering goes in both directions, we say that p and q are equivalent with respect to sublogic $\mathcal{O}$, written $p \sim_{\mathcal{O}} q$.[30]

By this account, $\langle\tau\rangle\neg\langle\mathsf{a}\rangle$ of Example 2.10 distinguishes P from Q. On the other hand, $\langle\tau\rangle \bigwedge\{\langle\mathsf{a}\rangle, \langle\mathsf{b}\rangle\}$ distinguishes Q from P. (The direction matters!) For instance, the sublogic $\{\langle\tau\rangle\langle\mathsf{a}\rangle, \langle\tau\rangle\langle\mathsf{b}\rangle\}$ preorders P and Q in both directions; so the two states are equivalent with respect to this logic.

Proposition 2.3. *Consider an arbitrary HML sublogic $\mathcal{O} \subseteq \mathsf{HML}$. Then, $\preceq_{\mathcal{O}}$ is a preorder, and $\sim_{\mathcal{O}}$ an equivalence relation.*[31]

Lemma 2.5. *Hennessy–Milner logic equivalence $\sim_{\mathsf{HML}}$ is a simulation relation.*[32]

Proof. Assume it were not. Then there would need to be $p \sim_{\mathsf{HML}} q$ with step $p \xrightarrow{\alpha} p'$, and no q' such that $q \xrightarrow{\alpha} q'$ and $p' \sim_{\mathsf{HML}} q'$. So there would need to be a distinguishing formula $\varphi_{q'}$ for each q' that q can reach by an α-step.

[27] primrec Hennessy_Milner_Logic.lts.satisfies

[28] abbreviation LTS_Semantics.lts.distinguishes

[29] definition LTS_Semantics.lts.preordered

[30] definition LTS_Semantics.lts.equivalent

[31] lemma LTS_Semantics.lts.equivalent_equiv

[32] lemma HML_Spectrum.lts.hml_equiv_sim

Consider the formula $\varphi_\alpha := \langle\alpha\rangle \bigwedge_{q' \in \mathrm{Der}(q,\alpha)} \varphi_{q'}$. It must be true for p and false for q, contradicting $p \sim_{\mathsf{HML}} q$. $\square$

Lemma 2.6 (HML Bisimulation Invariance). *If $p \in [\![\varphi]\!]$ and $p \sim_{\mathrm{B}} q$ then $q \in [\![\varphi]\!]$.*[33]

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

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

Combining the bisimulation invariance for one direction and that $\sim_{\mathsf{HML}}$ is a symmetric simulation (Proposition 2.3 and Lemma 2.5) for the other, we obtain that HML precisely characterizes bisimilarity:

Theorem 2.1 (Hennessy–Milner Theorem). *Bisimilarity and HML equivalence coincide, that is, $p \sim_{\mathrm{B}} q$ precisely if $p \sim_{\mathsf{HML}} q$.*[34]

Remark 2.2. In the standard presentation of the theorem, image finiteness of the transition system is assumed. This means that $\mathrm{Der}(p, \alpha)$ 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 2.5. But as our definition of HML (Definition 2.10) allows infinite conjunctions $\bigwedge_{i\in I} \ldots$, we do not need an assumption here. The implicit assumption is that the cardinality of index sets I can match that of $\mathrm{Der}(p, \alpha)$. The original proof by Hennessy & Milner (1980) uses binary conjunction $(\varphi_1 \land \varphi_2)$ 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 $\mathcal{O} \subseteq \mathsf{HML}$. There are three immediate big advantages to modal characterization:

Modal characterizations lead to *proper preorders and equivalences by design* due to Proposition 2.3. 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, modal characterizations can directly *unveil the hierarchy between preorders*, if defined cleverly, because of the following property.

Proposition 2.4. *If $\mathcal{O} \subseteq \mathcal{O}'$ then $p \preceq_{\mathcal{O}'} q$ implies $p \preceq_{\mathcal{O}} q$ for all p, q.*[35]

[33] lemma HML_Spectrum.lts.hml_bisim_invariant

[34] theorem HML_Spectrum.lts.Hennessy_Milner_theorem

[35] lemma LTS_Semantics.lts_semantics.preorder_contraposition

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

Thirdly, as a corollary of Proposition 2.4, modal characterizations *ensure equivalences to be abstractions of bisimilarity*, which is a sensible base notion of equivalence.[36]

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

[36] Among other things, bisimilarity checking has a better time complexity than other notions as will be discussed in Subsection 3.3.3.

2.3.4 Expressiveness and Distinctiveness

Proposition 2.4 actually is a weak version of another proposition about *distinctiveness* of logics.

Definition 2.13. 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 $[\![\varphi]\!] = [\![\varphi']\!]$. (The definition can also be read with regards to a fixed transition system $\mathcal{S}$.)[37] If the inclusion holds in both directions, $\mathcal{O}$ and $\mathcal{O}'$ are *equally expressive*.

[37] definition LTS_Semantics.lts_semantics.leq_expressive

Definition 2.14. 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.[38] If the inclusion holds in both directions, $\mathcal{O}$ and $\mathcal{O}'$ are *equally distinctive*.

[38] definition LTS_Semantics.lts_semantics.leq_distinctive

Lemma 2.7. *Subset relationship entails expressiveness entails distinctiveness.*

- *If $\mathcal{O} \subseteq \mathcal{O}'$, then $\mathcal{O}$ is less or equal in expressiveness to $\mathcal{O}'$.*[39]
- *If $\mathcal{O}$ is less or equal in expressiveness to $\mathcal{O}'$, then $\mathcal{O}$ is less or equal in distinctiveness to $\mathcal{O}'$.*[40]

[39] lemma LTS_Semantics.lts_semantics.subset_expressiveness

[40] lemma LTS_Semantics.lts_semantics.expressiveness_entails_distinctiveness

The other direction does not hold. For instance, $\mathsf{HML} \not\subseteq \mathsf{HML} \setminus \{\top\}$, 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 2.4 is thus:

Proposition 2.5. *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.*[41]

[41] lemma LTS_Semantics.lts_semantics.preorder_expressiveness_contraposition

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

Example 2.11. Consider $\mathcal{O}_{\lfloor \mathrm{B} \rfloor} \subseteq \mathsf{HML}$ described by the following grammar.

$$\begin{array}{rcl} \varphi^{\lfloor \mathrm{B} \rfloor} & ::= & \langle \alpha \rangle \bigwedge_{i \in I} \varphi_i^{\lfloor \mathrm{B} \rfloor} \\ & | & \neg \varphi^{\lfloor \mathrm{B} \rfloor} \end{array}$$

$\mathcal{O}_{\lfloor \mathrm{B} \rfloor}$ is a proper subset of HML. For instance, it lacks the observation $\langle \mathsf{a} \rangle \langle \mathsf{b} \rangle \top$. Due to the subset relation, $\mathcal{O}_{\lfloor \mathrm{B} \rfloor}$ 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 2.8. *$\mathcal{O}_{\lfloor \mathrm{B} \rfloor}$ and HML are equally distinctive.*[42]

[42] theorem HML_Spectrum.lts.hml_and_bisimulation_game_observations_equally_expressive

Proof. One direction is immediate from Lemma 2.7 as $\mathcal{O}_{\lfloor \mathrm{B} \rfloor} \subseteq \mathsf{HML}$.

For the other direction, we need to establish that for each $\varphi \in \mathsf{HML}$ distinguishing some p from some q, there is a $\varphi' \in \mathcal{O}_{\lfloor \mathrm{B} \rfloor}$ 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' with $p \xrightarrow{\alpha} p'$ distinguishing p' from all $q' \in \mathrm{Der}(q, \alpha)$. By induction hypothesis, there must be $\varphi'_{q'} \in \mathcal{O}_{\lfloor \mathrm{B} \rfloor}$ distinguishing p' from q' for each q'. Thus $\langle \alpha \rangle \bigwedge_{q' \in \mathrm{Der}(q,\alpha)} \varphi'_{q'} \in \mathcal{O}_{\lfloor \mathrm{B} \rfloor}$ 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 $\varphi'_i \in \mathcal{O}_{\lfloor \mathrm{B} \rfloor}$ distinguishes p from q.
- Case $\neg\varphi$ distinguishes p from q. Thus φ distinguishes q from p. By induction hypothesis, there is $\varphi' \in \mathcal{O}_{\lfloor \mathrm{B} \rfloor}$ distinguishing q from p. Accordingly, $\neg\varphi' \in \mathcal{O}_{\lfloor \mathrm{B} \rfloor}$ distinguishes p from q. □

The smaller bisimulation logic $\mathcal{O}_{\lfloor \mathrm{B} \rfloor}$ will become important again later in the next section (in Subsection 2.4.4).

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 2.15 (Reachability Game). A *reachability game* $\mathcal{G} = (G, G_{\mathrm{d}}, \rightarrowtail)$ is played on a directed graph consisting of

- a set of *game positions* G, partitioned into
  - *defender positions* $G_{\mathrm{d}} \subseteq G$
  - and *attacker positions* $G_{\mathrm{a}} := G \setminus G_{\mathrm{d}}$,
- and a set of *game moves* $\rightarrowtail \subseteq G \times G$.[43]

[43] locale Equivalence_Games.game

We denote by $\mathcal{G}(g_0)$ the game played from starting position $g_0 \in G$.

Definition 2.16 (Plays and Wins). We call the paths $g_0 g_1 ... \in G^\infty$ with $g_i \rightarrowtail g_{i+1}$ *plays* of $\mathcal{G}(g_0)$. They may be finite or infinite. The defender *wins* infinite plays. If a finite play $g_0 \ldots g_n \not\rightarrowtail$ is stuck, the stuck player loses: The defender wins if $g_n \in G_\mathrm{a}$, and the attacker wins if $g_n \in G_\mathrm{d}$.

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

Definition 2.17 (Strategies and Winning Strategies). An *attacker strategy* is a (partial) function mapping play fragments ending at attacker positions to next positions to move to, $s_\mathrm{a} \colon G^* G_\mathrm{a} \to G$, where $g_\mathrm{a} \rightarrowtail f_\mathrm{a}(\rho g_\mathrm{a})$ must hold for all ρg_a where s_a is defined.

A play $g_0 g_1 ... \in G^\infty$ is consistent with an attacker strategy s_a if, for all its prefixes $g_0 ... g_i$ ending in $g_i \in G_\mathrm{a}$, $g_{i+1} = s_\mathrm{a}(g_0 ... g_i)$.

Defender strategies are defined analogously, $s_\mathrm{d} \colon G^* G_\mathrm{d} \to G$.

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

Usually, we will focus on positional strategies, that is, strategies that only depend on the most recent position, which we will type $s_\mathrm{a} \colon G_\mathrm{a} \to G$ (or $s_\mathrm{d} \colon G_\mathrm{d} \to G$, respectively).

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

Definition 2.18 (Winning Region). The set $\mathsf{Win}_\mathrm{a} \subseteq G$ of all positions g where the attacker wins $\mathcal{G}(g)$ is called the *attacker winning region* of $\mathcal{G}$. The defender winning region Win_d is defined analogously.

Example 2.12 (A Simple Choice). Inspect the game in Figure 2.7, 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 $\mathsf{Win}_\mathrm{d} = \{(1), [2b]\}$ and the attacker wins $\mathsf{Win}_\mathrm{a} = \{[2a], (3)\}$.

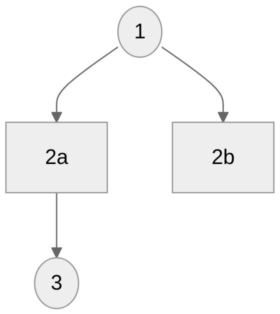


Figure 2.7: A simple game.

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 2.6 (Determinacy). *Reachability games are positionally determined, that is, for any game, each game position has exactly one winner: $G = \mathsf{Win}_\mathrm{a} \cup \mathsf{Win}_\mathrm{d}$ and $\mathsf{Win}_\mathrm{a} \cap \mathsf{Win}_\mathrm{d} = \emptyset$, and they can win using a positional strategy.*[44]

[44] This is just an instantiation of positional determinacy of parity games (Zielonka, 1998). Reachability games are the subclass of parity games only colored by 0.

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 2.13 (The $\leq$-Game). Consider the following recursive definition of a less-or-equal operator   on natural numbers in some functional programming language. (Consider `nat` to be defined as recursive data type `nat = 0 | 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 the left number 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 $\mathcal{G}_{\mathsf{leq}}$ consists of attacker positions $[n, m]$ and defender positions (n, m) for all $n, m \in \mathbb{N}$, connected by chains of moves:

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

$\mathcal{G}_{\mathsf{leq}}$ now characterizes $\leq$ on $\mathbb{N}$ in the sense that: The defender wins $\mathcal{G}_{\mathsf{leq}}$ 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 till they hit $[0, m - n]$ if $n \leq m$, or $(n - m, 0)$ otherwise.

$\mathcal{G}_{\mathsf{leq}}$ 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 (Definition 2.11) through a game. The defender wins precisely if the game starts from a state and formula such that the state satisfies the formula.

Definition 2.19 (HML Game). For a transition system $\mathcal{S} = (\mathcal{P}, \mathit{Act}, \rightarrow)$, the HML *game* $\mathcal{G}^{\mathcal{S}}_{\mathsf{HML}} = (G_{\mathsf{HML}}, G_{\mathrm{d}}, \rightarrowtail_{\mathsf{HML}})$ is played on $G_{\mathsf{HML}} = \mathcal{P} \times \mathsf{HML}$, where the defender controls observations and negated conjunctions, that is $(p, \langle\alpha\rangle\varphi) \in G_{\mathrm{d}}$ and $(p, \neg\bigwedge_{i \in I} \varphi_i) \in G_{\mathrm{d}}$ (for all φ, p, I), and the attacker controls the rest.

- The defender can perform the moves:

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

- and the attacker can move:

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

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

Example 2.14. The game of Example 2.12 is exactly the HML game $\mathcal{G}_{\mathsf{HML}}^{\mathcal{S}_{\mathsf{PQ}}}$ for formula $\langle\tau\rangle\neg\langle\mathsf{a}\rangle\top$ and state P of Example 2.10 with $(1) := (\mathsf{P}, \langle\tau\rangle\neg\langle\mathsf{a}\rangle\top)$, $[2\mathsf{a}] := (\mathsf{p_a}, \neg\langle\mathsf{a}\rangle\top)$, $[2\mathsf{b}] := (\mathsf{p_b}, \neg\langle\mathsf{a}\rangle\top)$, and $(3) := (\mathsf{p_1}, \neg\top)$.

The defender winning region $\mathsf{Win_d} = \{(\mathsf{P}, \langle\tau\rangle\neg\langle\mathsf{a}\rangle\top), (\mathsf{p_b}, \neg\langle\mathsf{a}\rangle\top)\}$ corresponds to the facts that $\mathsf{P} \in [\![\langle\tau\rangle\neg\langle\mathsf{a}\rangle\top]\!]$ and $\mathsf{p_b} \in [\![\neg\langle\mathsf{a}\rangle\top]\!]$.

As the technicalities are tangential to this thesis, we state the characterization result without proof:[45]

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

Lemma 2.9. *The defender wins $\mathcal{G}_{HML}^{\mathcal{S}}((p, \varphi))$ precisely if $p \in [\![\varphi]\!]$.*

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. This approach has been popularized by Stirling (1996).

Definition 2.20 (Bisimulation Game). For a transition system $\mathcal{S}$, the *bisimulation game* $\mathcal{G}_{\mathrm{B}}^{\mathcal{S}}$[46] is played on attack positions $G_{\mathrm{a}}^{\mathrm{B}} := \mathcal{P} \times \mathcal{P}$ and defense positions $G_{\mathrm{d}}^{\mathrm{B}} := \mathit{Act} \times \mathcal{P} \times \mathcal{P}$ with the following moves:

[46] locale Equivalence_Games.bisim_game

- The attacker may *swap sides*

$$[p, q] \quad \rightarrowtail_{\mathrm{B}} \quad [q, p],$$

- or *challenge simulation*

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

- and the defender *answers simulation challenges*

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

A schematic depiction of the game rules can be seen in Figure 2.8. From dashed nodes, the game proceeds analogously to the initial attacker position.

Example 2.15. Consider $\mathsf{p_{even}} \sim_{\mathrm{B}} \mathsf{p_{odd}}$ of Example 2.7. The bisimulation game on this system is given by Figure 2.9:

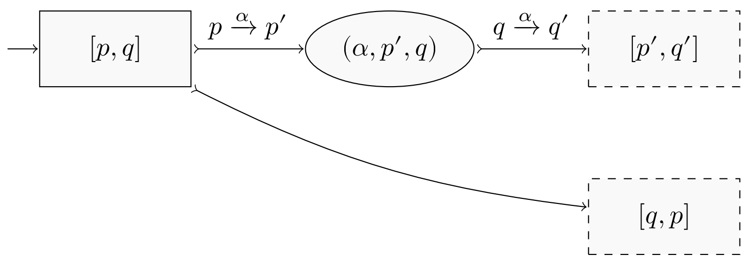


Figure 2.8: Game scheme of the bisimulation game (Definition 2.20).

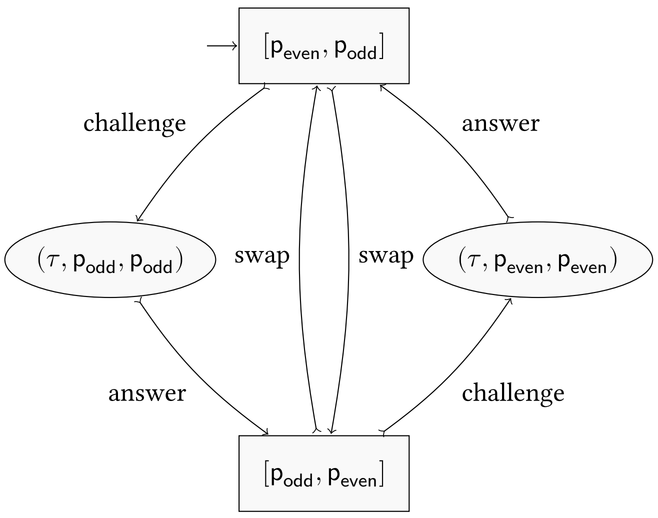

$[\mathsf{p_{even}}, \mathsf{p_{odd}}]$
challenge
answer
$(\tau, \mathsf{p_{odd}}, \mathsf{p_{odd}})$
swap
swap
$(\tau, \mathsf{p_{even}}, \mathsf{p_{even}})$
answer
challenge
$[\mathsf{p_{odd}}, \mathsf{p_{even}}]$

Figure 2.9: Bisimulation game under $\mathsf{p_{even}}, \mathsf{p_{odd}}$. 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 choses, 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 $[\mathsf{p_{even}}, \mathsf{p_{odd}}]$ and $[\mathsf{p_{odd}}, \mathsf{p_{even}}]$ corresponds to $\mathcal{R} := \{(\mathsf{p_{even}}, \mathsf{p_{odd}}), (\mathsf{p_{odd}}, \mathsf{p_{even}})\}$ being a bisimulation.

Example 2.16. Let us recall Example 2.6 of the "trolled philosophers," where we determined Q and T to be non-bisimilar. The bisimulation game graph for the system is depicted in Figure 2.10.

The attacker can win by moving $[\mathsf{Q}, \mathsf{T}] \rightarrowtail_{\mathrm{B}} [\mathsf{T}, \mathsf{Q}] \rightarrowtail_{\mathrm{B}} (\tau, \mathsf{q_3}, \mathsf{Q}) \rightarrowtail_{\mathrm{B}} [\mathsf{q_3}, \mathsf{q_{AB}}] \rightarrowtail_{\mathrm{B}} [\mathsf{q_{AB}}, \mathsf{q_3}] \rightarrowtail_{\mathrm{B}} (\mathsf{a}, \mathsf{q_1}, \mathsf{q_3}) \not\rightarrowtail_{\mathrm{B}}$. Along this sequence of positions, the defender never has a choice and is stuck in the end. The attacker exploits, that T can reach an early deadlock via $\mathsf{T} \xrightarrow{\tau} \mathsf{q_3}$.

Theorem 2.2 (Stirling's Game Characterization). *The defender wins the bisimulation game $\mathcal{G}_{\mathrm{B}}^{\mathcal{S}}$ starting at attacker position $[p, q]$ precisely if $p \sim_{\mathrm{B}} q$.*[47]

[47] theorem Equivalence_Games.bisim_game .bisim_game_characterization

Proof. Sketch for both directions:

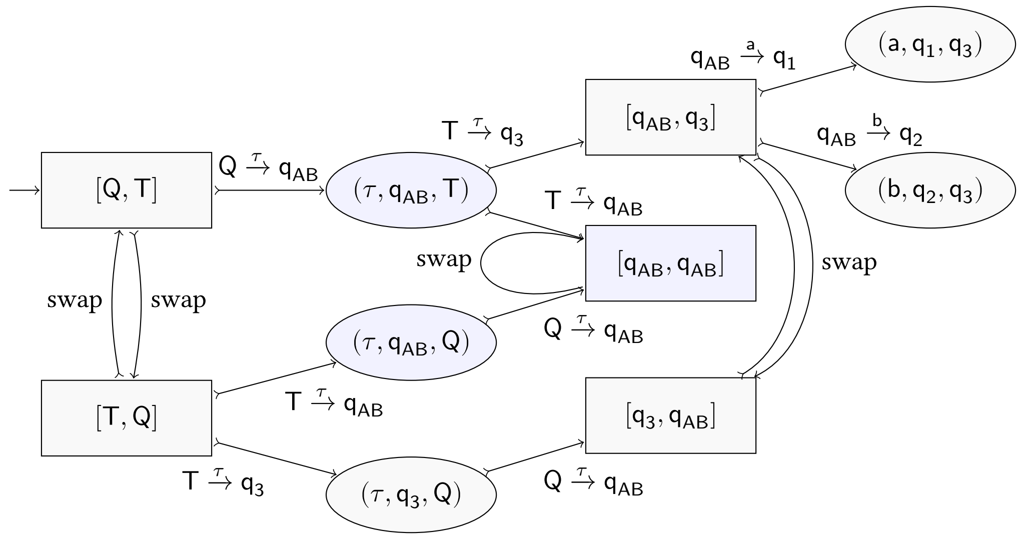

$$\begin{tikzcd}
 & & & \text{[q}_{AB}\text{, q}_3\text{]} \arrow[r, "q_{AB} \xrightarrow{a} q_1"] \arrow[dr, "q_{AB} \xrightarrow{b} q_2"] \arrow[dd, bend left, "\text{swap}"] & (a, q_1, q_3) \\
 {[Q, T]} \arrow[r, "Q \xrightarrow{\tau} q_{AB}"] \arrow[d, bend right, "\text{swap}"'] & (\tau, q_{AB}, T) \arrow[urr, "T \xrightarrow{\tau} q_3"] \arrow[rr, "T \xrightarrow{\tau} q_{AB}"] & & {[q_{AB}, q_{AB}]} \arrow[loop left, "\text{swap}"] & (b, q_2, q_3) \\
 {[T, Q]} \arrow[u, bend right, "\text{swap}"'] \arrow[r, "T \xrightarrow{\tau} q_{AB}"] \arrow[dr, "T \xrightarrow{\tau} q_3"'] & (\tau, q_{AB}, Q) \arrow[urr, "Q \xrightarrow{\tau} q_{AB}"] & & {[q_3, q_{AB}]} \arrow[uu, bend right, "\text{swap}"'] & \\
 & (\tau, q_3, Q) \arrow[urr, "Q \xrightarrow{\tau} q_{AB}"] & & &
\end{tikzcd}$$

Figure 2.10: Bisimulation game on non-bisimilar states Q and T of Example 2.16. Moves are labeled by their justification. Defender-won positions are tinted blue.

- If $\mathcal{R}$ is a symmetric simulation with $(p, q) \in \mathcal{R}$, then the following positional defender strategy is well-defined and winning from $[p, q]$:[48]

$$s(\alpha, p', q) := [p', \text{choose } q' .\, (p', q') \in \mathcal{R} \wedge q \xrightarrow{\alpha} q'].$$

- If there is a positional defender strategy s winning from $[p, q]$, then the following relation $\mathcal{R}_s$ with $(p_0, q_0) \in \mathcal{R}_s$ is a symmetric simulation:[49]

$$\mathcal{R}_s := \{(p, q) \mid \text{there is a play } (p_0, q_0), \ldots, (p, q) \text{ following strategy } s\}.$$

□

[48] lemma Equivalence_Games.bisim_game .bisim_implies_defender_winning_strategy

[49] lemma Equivalence_Games.bisim_game .defender_winning_strategy_implies_bisim

Remark 2.3. 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 2.20. Figure 2.11 shows a screenshot of said game. It can be played on https://www.concurrency-theory.org/rvg-game/.

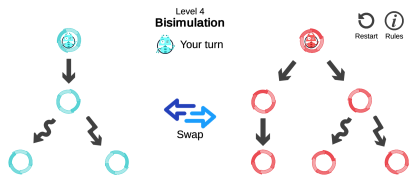


Figure 2.11: Screenshot of Peacock's bisimulation computer game.

Example 2.17. If one plays the bisimulation game of Definition 2.20 without the swapping moves, it will characterize the simulation preorder.

Consider the family of processes N_n with $n \in \mathbb{N}$. Define $\mathsf{N}_0 := \mathbf{0}$ and $\mathsf{N}_{n+1} := \mathsf{a}.\mathsf{N}_n$. Then, the simulation game played from $[\mathsf{N}_n, \mathsf{N}_m]$ is isomorphic to the $\leq$-game $\mathcal{G}_{\mathsf{leq}}$ from Example 2.13.

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

2.4.4 How Bisimulation Game and HML Are Connected

Let us pause to think how bisimulation game and Hennessy–Milner logic connect.

You might have wondered why we even need a dedicated bisimulation game. The Hennessy–Milner theorem implies that we could directly use the HML game of Definition 2.19 to decide bisimilarity:

Definition 2.21 (Naive Bisimulation Game). We extend the Definition 2.19 by the following prefix:

1. To challenge $[p, q]$, the attacker picks a formula $\varphi \in \mathsf{HML}$ (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 2.19.

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

The proper bisimulation game of Definition 2.20, on the other hand, is *finite for finite transition systems*. Therefore, it induces decision procedures.

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 $\mathcal{O}_{\lfloor \mathrm{B} \rfloor} \subseteq \mathsf{HML}$ of Example 2.11. To this end, we will show that attacker's winning strategies imply distinguishing formulas, and that a distinguishing formula from $\mathcal{O}_{\lfloor \mathrm{B} \rfloor}$ certifies the existence of winning attacker strategies.

Example 2.18. Let us illustrate how to derive distinguishing formulas using the game of Example 2.16.

Recall that the attacker wins by moving $[\mathsf{Q}, \mathsf{T}] \rightarrowtail_{\mathrm{B}} [\mathsf{T}, \mathsf{Q}] \rightarrowtail_{\mathrm{B}} (\tau, \mathsf{q_3}, \mathsf{Q}) \rightarrowtail_{\mathrm{B}} [\mathsf{q_3}, \mathsf{q_{AB}}] \rightarrowtail_{\mathrm{B}} [\mathsf{q_{AB}}, \mathsf{q_3}] \rightarrowtail_{\mathrm{B}} (\mathsf{a}, \mathsf{q_1}, \mathsf{q_3}) \not\rightarrowtail_{\mathrm{B}}$. In Figure 2.12, we label the game nodes with the (sub-)formulas this strategy corresponds to. 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 $\neg\langle\tau\rangle \bigwedge\{\neg\langle \mathsf{a}\rangle\top\} \in \mathcal{O}_{\lfloor \mathrm{B} \rfloor}$.

More generally, the following lemma explains the construction of distinguishing formulas from attacker strategies:

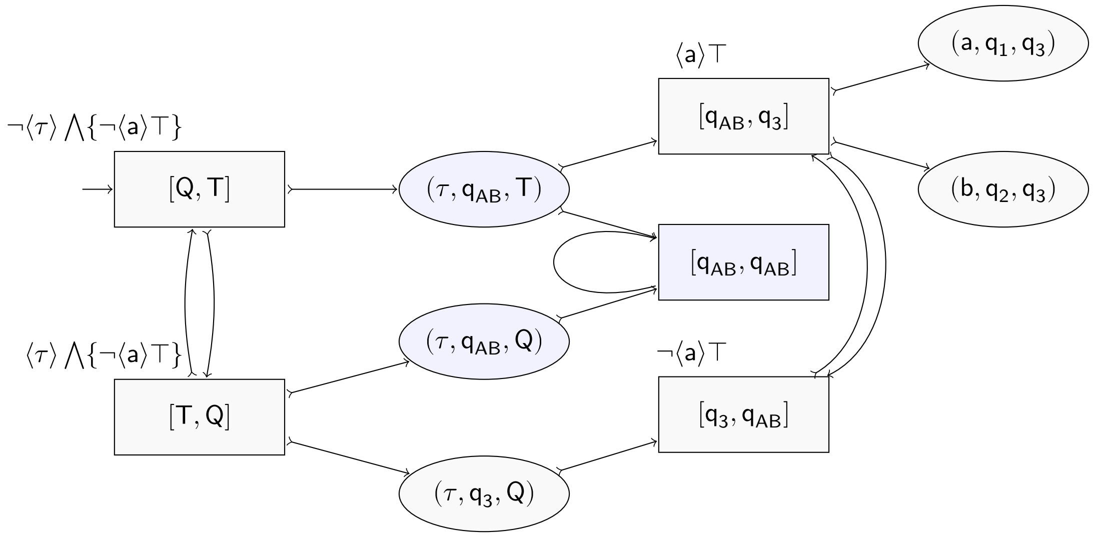


Figure 2.12: The bisimulation game of Example 2.18 with attacker formulas.

Lemma 2.10. *Let s be a positional winning strategy for the attacker on $\mathcal{G}_{\mathrm{B}}$ from $[p, q]$. Construct formulas recursively from game positions, $\varphi_s(g)$, as follows:*

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

Then $\varphi_s([p, q])$ is well-defined and distinguishes p from q. Also, $\varphi_s([p, q]) \in \mathcal{O}_{\lfloor\mathrm{B}\rfloor}$.

Proof.

1. The given construction is well-defined as s must induce a well-founded order on game positions in order to be attacker-winning.
2. The distinction can be derived by induction on the construction of $\varphi_s([p, q])$. TODO □

Lemma 2.11. *If $\varphi \in \mathcal{O}_{\lfloor\mathrm{B}\rfloor}$ distinguishes p from q, then the attacker wins from $[p, q]$.*

Proof. By induction on the derivation of $\varphi \in \mathcal{O}_{\lfloor\mathrm{B}\rfloor}$ according to the definition from Example 2.11 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' such that $p \xrightarrow{\alpha} p'$ and that $\bigwedge_{i\in I} \varphi_i$ distinguishes p' from every $q' \in \mathrm{Der}(q, \alpha)$. That is, for each $q' \in \mathrm{Der}(q, \alpha)$, at least one $\varphi_i \in \mathcal{O}_{\lfloor\mathrm{B}\rfloor}$ 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. □

Lemma 2.10 and Lemma 2.11, together with the fact that $\mathcal{O}_{\lfloor \mathrm{B} \rfloor}$ and HML are equally distinctive (Lemma 2.8), yield:

Theorem 2.3. *The attacker wins $\mathcal{G}_{\mathrm{B}}$ from $[p, q]$ precisely if there is $\varphi \in$ HML distinguishing p from q.*

Of course, we already knew this! Theorem 2.3 is just another way of gluing together the Hennessy–Milner theorem on bisimulation (Theorem 2.1) and the Stirling's bisimulation game characterization (Theorem 2.2), modulo the determinacy of games. We thus have added the last arrow on the right side of Figure 2.1.

2.4.5 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. 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 $|{\rightarrowtail}|$.

```
def compute_winning_region($\mathcal{G} = (G, G_{\mathrm{d}}, \rightarrowtail)$):
  defender_options := $[g \mapsto n \mid g \in G \land n = |\{g' \mid g \rightarrowtail g'\}|]$
  attacker_win := $\emptyset$
  todo := $\{g \in G_{\mathrm{d}} \mid$ defender_options$[g] = 0\}$
  while todo $\neq \emptyset$:
    g := some todo
    todo := todo $\setminus$ {g}
    if g $\notin$ attacker_win:
      attacker_win := attacker_win $\cup$ {g}
      for $\mathsf{g_p} \in (\cdot \rightarrowtail \mathsf{g})$:
        defender_options[$\mathsf{g_p}$] := defender_options[$\mathsf{g_p}$] $- 1$
        if $\mathsf{g_p} \in G_{\mathrm{a}} \lor$ defender_options[$\mathsf{g_p}$] $= 0$:
          todo := todo $\cup$ {$\mathsf{g_p}$}
  $\mathsf{Win_a}$ := attacker_win
  return $\mathsf{Win_a}$
```

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

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.[50]

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

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 $\mathsf{Win}_\mathrm{a} = \{[\mathsf{2a}], (3)\}$ of the game of Example 2.11.

Table 2.1: Solving the game of Example 2.11

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

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

Proposition 2.7. *Given a finite reachability game $\mathcal{G} = (G, G_\mathrm{d}, \rightarrowtail)$, the attacker's winning region* Win_a *(and* Win_d *as well) can be computed in $O(|{\rightarrowtail}|)$ time and $O(|G|)$ space.*

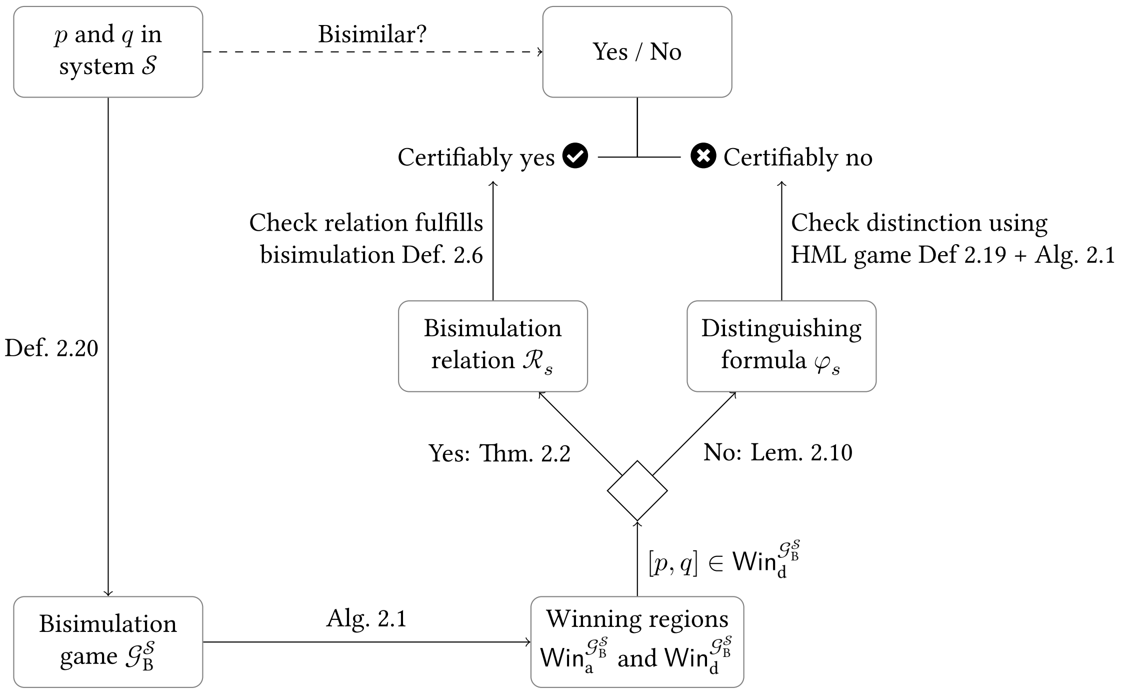
2.5 Discussion

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.

In particular, 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 2.13 describes how to not only answer the question whether two states p and q are bisimilar, but how to also either provide a bisimulation 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 witnesses 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 (Note 1). This whole framework draws heavily from Stirling (1996).



p and q in system $\mathcal{S}$
Bisimilar?
Yes / No
Certifiably yes ✓
✗ Certifiably no
Check relation fulfills bisimulation Def. 2.6
Check distinction using HML game Def 2.19 + Alg. 2.1
Def. 2.20
Bisimulation relation $\mathcal{R}_s$
Distinguishing formula φ_s
Yes: Thm. 2.2
No: Lem. 2.10
$[p, q] \in \mathsf{Win}_\mathrm{d}^{\mathcal{G}_\mathrm{B}^\mathcal{S}}$
Bisimulation game $\mathcal{G}_\mathrm{B}^\mathcal{S}$
Alg. 2.1
Winning regions $\mathsf{Win}_\mathrm{a}^{\mathcal{G}_\mathrm{B}^\mathcal{S}}$ and $\mathsf{Win}_\mathrm{d}^{\mathcal{G}_\mathrm{B}^\mathcal{S}}$

Figure 2.13: Checking bisimilarity and providing certificates.

We have disregarded the topic of axiomatic characterizations for behavioral equivalences. In these, sets of equations on process algebra terms (such as CCS) define behavioral congruences. Thus, they tend to be coupled to specific process languages and lack the genericity we pursue in starting out from transition systems.

We have observed that behavioral equivalences can be arranged in hierarchies and that these hierarchies could be handled nicely using modal characterizations to rank distinguishing powers (Subsection 2.3.3). We have also seen how to link game rules to productions in a language of potentially distinguishing formulas (Subsection 2.4.4). 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 (Note 2), to generalize the framework of Figure 2.13 for whole spectra of equivalences.

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

3 Context: The Spectrum of Equivalences

We will now take a deeper dive into 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 Glabbeek (1990), the so-called "strong spectrum." However, we will order the equivalences using the approach of parameterized *notions of observability* from the later Glabbeek (1993).

The first core idea of this chapter is:

> Idea 3: 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 will quickly run into the second core idea:

> Idea 4: We have a Spectroscopy problem
>
> We can ask what equivalences from a spectrum are the most fitting to relate two states.

Section 3.3 will define the 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 Subsection 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 do become.

3.1.1 Understanding the Equivalence Hierarchy through Modal Logics

As promised, we revisit the hierarchy between bisimilarity, similarity, and trace equivalence of Subsection 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 $\mathcal{O}_{\lfloor B \rfloor}$ in Example 2.11.

Definition 3.1. 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.

$$\begin{array}{lcl} \varphi^T & ::= & \langle\alpha\rangle\varphi^T \quad | \quad \top \\ \varphi^S & ::= & \langle\alpha\rangle\varphi^S \quad | \quad \bigwedge_{i\in I}\langle\alpha_i\rangle\varphi^S_i \end{array}$$

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

Lemma 3.1. $p \preceq_T q$ *precisely if* $p \preceq_{\mathcal{O}_T} q$.[51]

Lemma 3.2. $p \preceq_S q$ *precisely if* $p \preceq_{\mathcal{O}_S} q$.[52]

[51] theorem HML_Spectrum.lts.observations_traces_characterizes_trace_preorder

[52] theorem HML_Spectrum.lts.observations_simulation_characterize_simulation_preorder

Clearly, $\mathcal{O}_T \subset \mathcal{O}_S \subset$ HML. So, Proposition 2.4 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 Subsection 2.2.3.[53]

[53] Taking the two previous lemmas together with Theorem 2.1 for $\sim_B$ and HML.

While the relational definitions for (bi-)similarity of Definition 2.7 and the denotational definition of trace equivalence Definition 2.5 live in different worlds, the two equivalences become naturally comparable in the modal realm.

The modal view also reveals an intuitive hierarchy of "testing scenarios" for the equivalences, framed as black box tests:

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 …

3.1.2 Incomparabilities

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 $\mathsf{Failures}(p)$ is recursively defined as

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

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

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

Definition 3.3. We define failure observations $\mathcal{O}_\mathrm{F}$ by the grammar:

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

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

The distinguishing failure $(\tau, \{\mathsf{a}\})$ can be as $\langle\tau\rangle \bigwedge\{\neg\langle\mathsf{a}\rangle\top\} \in \mathcal{O}_\mathrm{F}$ in HML. The formula distinguishes P from Q on Figure 2.3, but $\mathsf{P} \preceq_\mathrm{S} \mathsf{Q}$ (cf. Example 2.5). On the other hand, no failure from $\mathcal{O}_\mathrm{F}$, distinguishes Q from P, but $\mathsf{Q} \not\preceq_\mathrm{S} \mathsf{P}$.

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 3.1. 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.

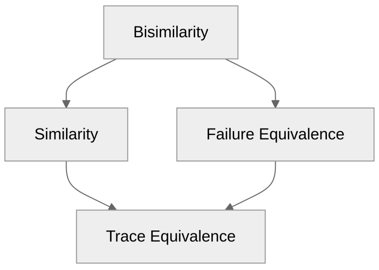


Figure 3.1: Preorder/equivalence hierarchy.

3.1.3 Lattices

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

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 $b' \in B'$.[54]

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 $b_1 \sqcup b_2 = \sup\{b_1, b_2\}$.

[54] Note that not necessarily $b \in B'$!

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 $\sqcap$-complete or $\sqcup$-complete if only one of the two is true.

We call 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*.

Example 3.1 (Subset Lattice). Given any set B, its subsets and 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 $B' \subseteq B$. $\emptyset \in 2^B$ is the least element, $B \in 2^B$ is itself the greatest bound.

Consider the subset lattice over $B = \{1, 2, 3\}$. It is "cube-like," as can be seen in its Hasse diagram in Figure 3.2. 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 $\emptyset \subset \{1\} \subset \{1, 2\} \subset B$. $\{\{1\}, \{2\}, \{3\}\}$ 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.

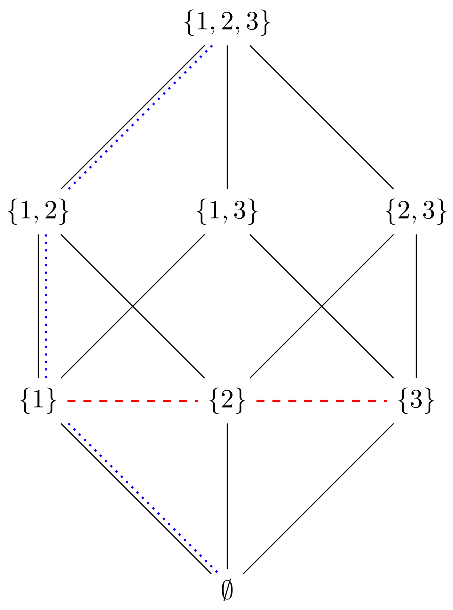


Figure 3.2: Lattice of subsets from Example 3.1.

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

For instance, pairs of natural numbers, $(\mathbb{N}^2, \leq)$, form a lattice, as visualized in Figure 3.3. It is $\sqcap$-complete, that is, for any set from $\mathbb{N}^2$, one can pick a greatest lower bound. However, the lattice is not $\sqcup$-complete: For instance, the set $\mathbb{N} \times \{0\}$ has no upper bound. If we take the natural numbers extended with an upper bound ∞, $\mathbb{N}_\infty$, as basis, then $(\mathbb{N}^2_\infty, \leq)$ forms a complete lattice.

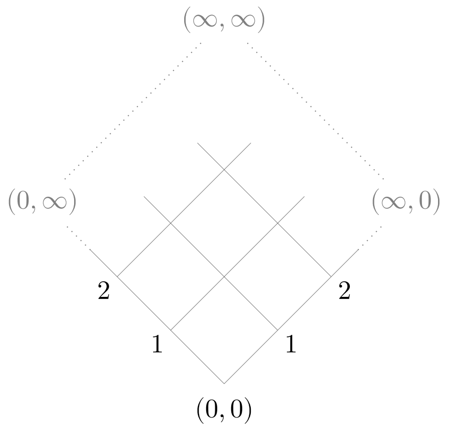


Figure 3.3: Visualization of the infinitary grid of $\mathbb{N}^2$ (and in gray, $\mathbb{N}^2_\infty$) in Example 3.2.

3.2 The Linear-Time–Branching-Time Spectrum

Van Glabbeek's two papers on the "linear-time–branching-time spectrum" (1993, 2001) 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*.[55] We will introduce a similar construction: at first, in a generic form, then for the strong spectrum of Glabbeek (2001).

[55] In particular, the weak spectrum (Glabbeek, 1993) makes this really formal.

3.2.1 Spectra as Observability Lattices

As we will discuss various equivalence spectra (van Glabbeek (1993, 2001) 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 $\mathcal{O}_N : 2^{\mathsf{HML}}$ for $N \in \mathbf{N}$.

$\mathcal{O}_{N \in notions}$ 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 3.1.

Example 3.3. Consider the notions

$$\mathbf{N}^{\mathsf{simple}} := 2^{\{(\wedge), (\neg)\}},$$

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

$$\begin{array}{lcll} \varphi^N & ::= & \top & \\ & | & \langle \alpha \rangle \varphi^N & \\ & | & \bigwedge_{i \in I} \varphi_i^N & \text{if } (\wedge) \in N \\ & | & \bigwedge_{i \in I} \neg \langle \alpha_i \rangle \top & \text{if } (\neg) \in N \\ & | & \neg \varphi^N & \text{if } \{(\wedge), (\neg)\} = N. \end{array}$$

Clearly, $N \subseteq M$ implies $\mathcal{O}_N \subseteq \mathcal{O}_M$ for $N, M \in \mathbf{N}^{\mathsf{simple}}$. We obtain the diamond hierarchy of Figure 3.4. It matches the diamond of Figure 3.1, but this time, the hierarchy is an effect of the ordered notions.

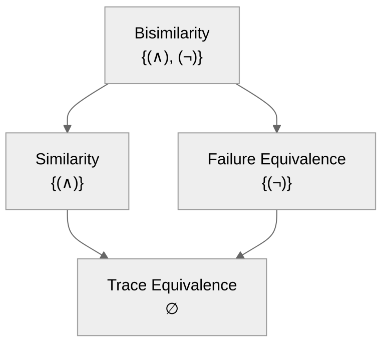


Figure 3.4: Hierarchy of simple notions of equivalence.

While the incomparable languages of Subsection 3.1.2 form no lattice, e.g. $\mathcal{O}_{\mathrm{S}} \cup \mathcal{O}_{\mathrm{F}} \neq \mathsf{HML}$, the *notions* of the present Example 3.3 do form a lattice, as $\{(\wedge)\} \cup \{(\neg)\} = \{(\wedge), (\neg)\}$. 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 $\sqcap$-complete 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 is defined as

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

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: $\mathsf{expr}(\langle \tau \rangle \langle \mathsf{a} \rangle) = \emptyset \subset \{(\neg)\} = \mathsf{expr}(\langle \tau \rangle \bigwedge \{\neg \langle \mathsf{a} \rangle\})$, which captures that we need a strictly smaller part of the grammar to construct the trace formula.

3.2.2 The Strong Notions of Observability

Now we can approach the *strong spectrum*. We will encode its notions as a $\mathbb{N}_\infty$-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. Formula modal depth of observations.
2. Formula nesting depth of conjunctions.
3. Maximal modal depth of deepest positive clauses in conjunctions.
4. Maximal modal depth of the other positive clauses in conjunctions.
5. Maximal modal depth of negative clauses in conjunctions.
6. Formula nesting depth of negations.

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

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

ordered by pointwise comparison of vector components, and the family of strong observation languages $\mathcal{O}^{\mathsf{strong}}_{N \in \mathbf{N}^{\mathsf{strong}}}$ given by the parameterized grammar starting from φ^N:[56]

[56] primrec Priced_HML.formula_of_price

$$\begin{array}{lcl}
\varphi^N & ::= & \top \\
 & \mid & \langle\alpha\rangle\varphi^{N-\hat{\mathbf{e}}_1} \\
 & \mid & \bigwedge\{\psi^{N-\hat{\mathbf{e}}_2}, \psi^{(N-\hat{\mathbf{e}}_2)\sqcap(\infty,\infty,N_4,\infty,\infty,\infty)}, \psi^{(N-\hat{\mathbf{e}}_2)\sqcap(\infty,\infty,N_4,\infty,\infty,\infty)}, \ldots\} \\
\psi^N & ::= & \langle\alpha\rangle\varphi^{(N\sqcap(N_3,\infty,\infty,\infty,\infty,\infty))-\hat{\mathbf{e}}_1} \\
 & \mid & \neg\langle\alpha\rangle\varphi^{(N\sqcap(N_5,\infty,\infty,\infty,\infty,\infty))-\hat{\mathbf{e}}_1-\hat{\mathbf{e}}_6}
\end{array}$$

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

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

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

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 clauses. A visualization for how $\mathsf{expr}^{\mathsf{strong}}(\langle\tau\rangle\bigwedge\{\neg\langle\mathsf{a}\rangle\top\}) = (2,1,0,0,1,1)$ (according to Definition 3.6) comes together is given in the upcoming Figure 3.5. Formally, the reason is that the following derivation is optimal:

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

If we reexamine the grammar of $\mathcal{O}_F$ in Definition 3.3, we notice that the formulas it can produce match those of $\mathcal{O}^{\text{strong}}_{(\infty,1,0,0,1,1)}$ of the strong spectrum in Definition 3.7.

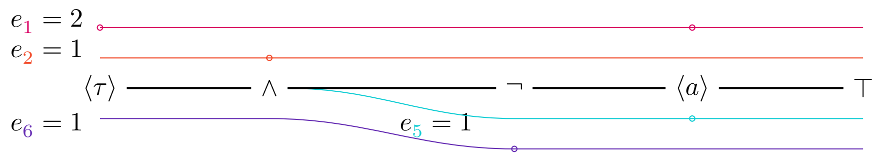


Figure 3.5: Pricing formula $\langle\tau\rangle \bigwedge\{\neg\langle\mathsf{a}\rangle\top\}$ with syntactic expressiveness $e = (2, 1, 0, 0, 1, 1)$.

An example for the pricing of a more complex tree-like formula is given in Figure 3.6.

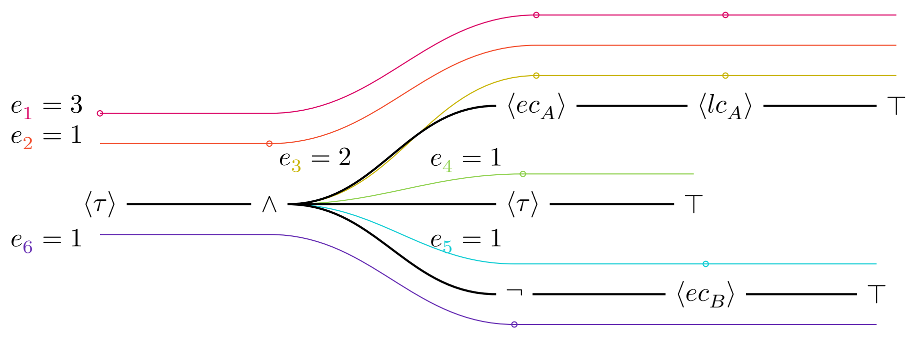


Figure 3.6: Pricing formula $\langle\tau\rangle \bigwedge\{\langle ec_A\rangle\langle lc_A\rangle\top, \langle\tau\rangle\top, \neg\langle ec_B\rangle\top\}$ with syntactic expressiveness $e = (3, 1, 2, 1, 1, 1)$.

Lemma 3.3. *The strong spectrum of Definition 3.7 contains the notions of behavioral equivalence we have discussed so far.*

1. *The observation language $\mathcal{O}^{\text{strong}}_{(\infty,0,0,0,0,0)}$ exactly matches the characterization of traces $\mathcal{O}_T$ from Definition 3.1 and thus characterizes trace preorder.*[57]
2. *The observation language $\mathcal{O}^{\text{strong}}_{(\infty,\infty,\infty,\infty,0,0)}$ exactly matches the characterization of simulation observations $\mathcal{O}_S$ from Definition 3.1 and thus characterizes simulation.*[58]
3. *The observation language $\mathcal{O}^{\text{strong}}_{(\infty,\infty,\infty,\infty,\infty,\infty)}$ matches HML in distinctiveness and thus characterizes bisimilarity.*

[57] theorem Priced_Spectrum.lts.traces_priced_characterization

[58] theorem Priced_Spectrum.lts.simulation_priced_characterization

4. *The observation language $\mathcal{O}^{\mathsf{strong}}_{(\infty,1,0,0,1,1)}$ exactly matches failure observations $\mathcal{O}_{\mathrm{F}}$ of Definition 3.3.*

Proof. The only non-trivial case is (3) for bisimilarity:

Observe that $\mathcal{O}_{\lfloor \mathrm{B} \rfloor} \subseteq \mathcal{O}^{\mathsf{strong}}_{(\infty,\infty,\infty,\infty,\infty,\infty)}$ by examining its grammar in Example 2.11. As $\mathcal{O}_{\lfloor \mathrm{B} \rfloor}$ already has complete HML-distinctiveness by Lemma 2.8, so must its superlogic $\mathcal{O}^{\mathsf{strong}}_{(\infty,\infty,\infty,\infty,\infty,\infty)}$. □

So far, we have only established that the six-dimensional spectrum also covers the notions that 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 3.7.[59]

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

For the rest of the thesis, we will take the equivalences as defined by the coordinates as canonical. Lemma 3.3 has established that the coordinates of traces, simulation, failures, and bisimulation match the definitions one commonly finds in the literature. To establish the same for the other notions, would mean a lot of (repetitive) definitions, which mostly reproduce van Glabbeek's survey (2001). Mattes (2024) proves in Isabelle/HOL that the coordinate system of the conference version of this spectrum (Bisping, 2023a) match distinctiveness of the modal characterizations in (Glabbeek, 2001).

⚠ Warning

I might add such a marathon if there's strong popular demand.

There are a few standard questions that come to mind for people who are familiar with the various spectra of equivalences when seeing Figure 3.7. The following remarks address these points.

Remark 3.1 (Selection of Notions). At the core, we treat the same notions as Glabbeek (2001). But we feature a slightly more modern selection.

Our spectrum additionally includes strong versions of *impossible futures* (Voorhoeve & Mauw (2001)) and *revivals* (Roscoe (2009)) as equivalences whose relevance has only been noted after the publication of 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

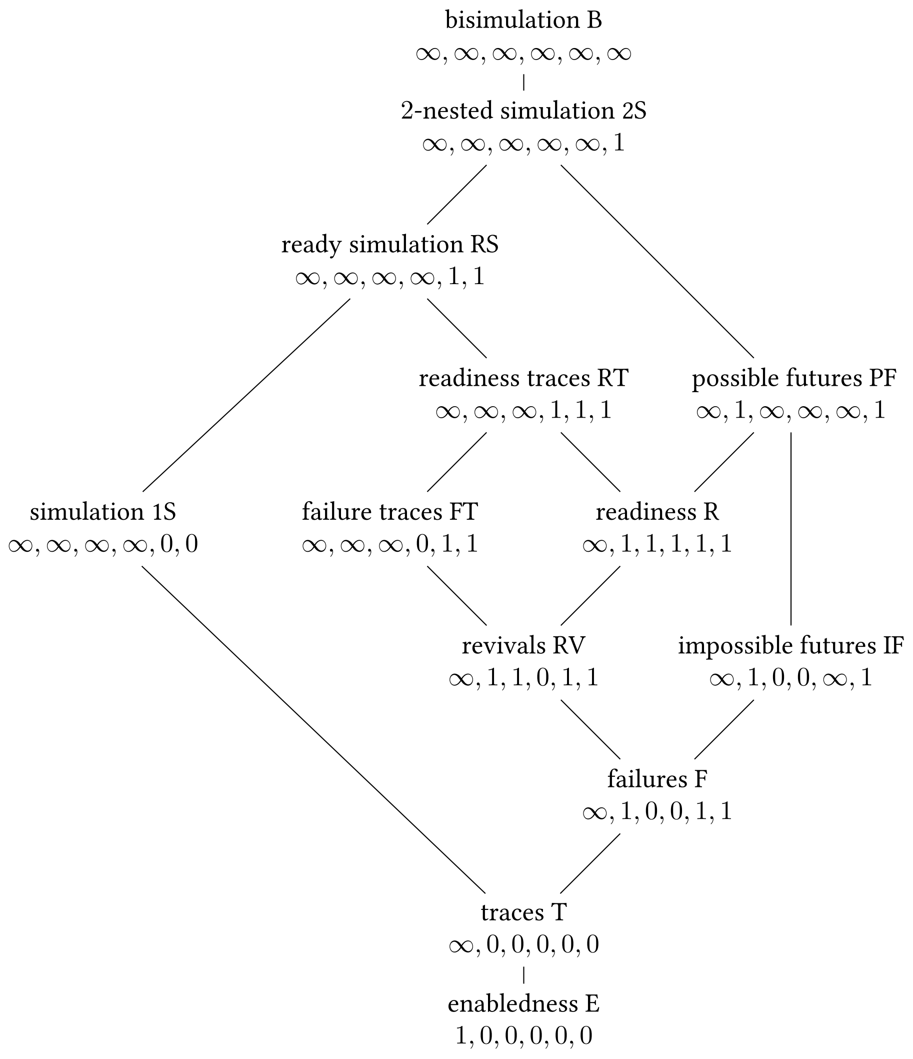


Figure 3.7: Hierarchy of common equivalences/preorders ordered by observability coordinates.

different HML grammar, featuring exhaustive conjunctions $\bigwedge_{a \in \mathit{Act}} \varphi_a$, where the φ_a are deactivated actions for completed traces, and more complex trees for possible worlds.

Remark 3.2 (Ambiguous Coordinates). For many of the logics in Figure 3.7, 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 $\mathcal{O}_N^{\mathsf{strong}} = \mathcal{O}_{\mathrm{T}}^{\mathsf{strong}}$.

Indeed, Figure 3.7 always selects the least coordinate to characterize a sublogic, which ensures that domination of coordinates in the figure and entailment between behavioral preorders coincide.

Remark 3.3 (Other Coordinates). We have singled out a handful of coordinates. Many other coordinates will still correspond to distinct equivalences. For instance, we could consider $N^{2\mathrm{T}} = (2, 0, 0, 0, 0, 0)$, 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.

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

- k-step bisimilarity: $(k, \infty, \infty, \infty, \infty, \infty)$ is a k-observation approximation of bisimilarity that sometimes appears in proofs.[60]
- 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.

[60] TODO: look up example

Remark 3.4 ((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 3.7 correspond precisely to those without superscripts in the infinitary linear-time–branching-time spectrum of Glabbeek (2001, Figure 9).

3.2.4 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 $\mathcal{O}_{\mathrm{S}}^{\mathsf{strong}} \cup \mathcal{O}_{\mathrm{F}}^{\mathsf{strong}} \neq \mathcal{O}_{\mathrm{RS}}^{\mathsf{strong}}$ and this makes a difference:

Example 3.5. Consider the CCS processes $\mathsf{a}.(\mathsf{a}.\mathsf{b} + \mathsf{a})$ and $\mathsf{a}.\mathsf{a}.\mathsf{b} + \mathsf{a}.\mathsf{a}$. They cannot be told apart by $\mathcal{O}_\mathrm{S}^{\mathsf{strong}}$ or $\mathcal{O}_\mathrm{F}^{\mathsf{strong}}$ and thus are simulation and failure equivalent (and moreover even ready-trace equivalent).

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

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

$$\sim_\mathrm{S} \cap \sim_\mathrm{F} \not\subseteq \sim_\mathrm{RS}.$$

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 to simple, one has to ask: “Which equivalences relate two processes?”

This 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.

Definition 3.9 (Spectroscopy Problem). In the context of a transition system $\mathcal{S}$ and a spectrum $(\mathbf{N}, \leq, \mathcal{O}_{N \in \mathbf{N}})$, the *spectroscopy problem* asks:

Input States p and q.
Output Set of notions $\mathbf{N}_{p,q} \subseteq \mathbf{N}$, such that $p \preceq_{\mathcal{O}_N} q$ for each $N \in \mathbf{N}_{p,q}$.

Example 3.6. For the “trolled philosophers” of Example 2.6, we have determined that the systems are simulation-preordered, but not bisimilar, that is, $\mathsf{Q} \preceq_\mathrm{S} \mathsf{T}$, but $\mathsf{Q} \not\preceq_\mathrm{B} \mathsf{T}$. The first fact implies $\mathsf{Q} \preceq_\mathrm{T} \mathsf{T}$.

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

The solution to the spectroscopy problem on Q and T is $\mathbf{N}_{\mathsf{Q},\mathsf{T}}^{\mathsf{strong}} = \{N \in \mathbf{N}^{\mathsf{strong}} \mid N \not\leq (2,2,0,0,2,2)\}$. A minimal formula to distinguish Q from T with this coordinate would be $\bigwedge \{\neg \langle \tau \rangle \bigwedge \{\neg \langle \mathsf{a} \rangle \top\}\}$. (The following chapters will reveal how to reliably arrive at this knowledge, in particular, the minimality.)

Because the coordinate of 2-nested simulation, $2S = (\infty, \infty, \infty, \infty, \infty, 1)$ is not less or equal to $(2, 2, 0, 0, 2, 2)$, we arrive at $\mathsf{Q} \preceq_{2S} \mathsf{T}$, which implies *all* preorders of Figure 3.7 except for bisimilarity.

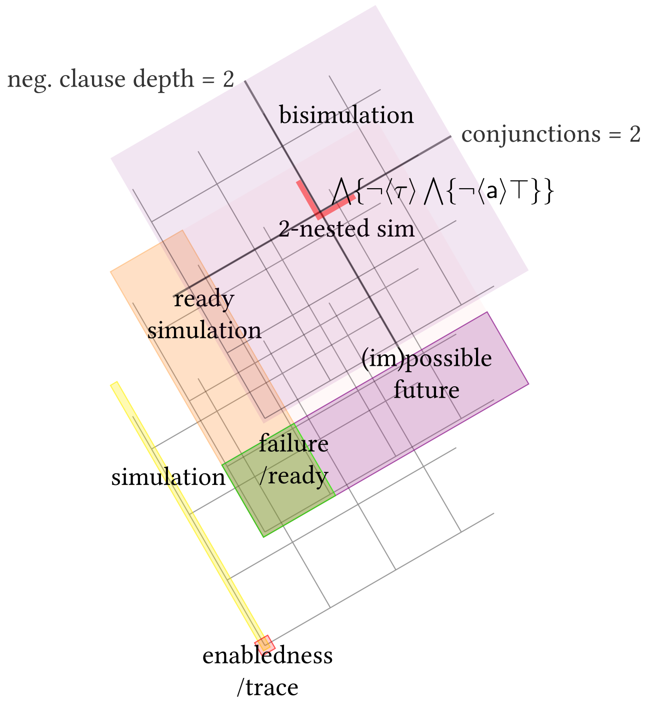


Figure 3.8: Cross section of the strong spectrum showing the dimensions conjunctions, negative clause depth, and negation depth. All preorders below the red mark at $(2, 2, 2)$ hold in Example 3.6.

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

Technically, it is convenient to not compute $\mathbf{N}_{p,q}$ directly. Rather we aim to construct the *Pareto front* of minimal notions that do not hold, $\mathrm{Min}(\mathbf{N} \setminus \mathbf{N}_{p,q})$. The Pareto front serves as a unique representation, from which $\mathbf{N}_{p,q}$ can be constructed as complement of the upwards closure $\mathbf{N} \setminus \uparrow \mathrm{Min}(\mathbf{N} \setminus \mathbf{N}_{p,q})$ (where $\uparrow B := \{b \mid \exists b' \in B.\, b' \leq b\}$ is the upwards closure for B subset of some partially-ordered set). Clearly, 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 $\mathrm{Min}(\mathbf{N} \setminus \mathbf{N}_{p,q})$ must be finite in size (Kruskal, 1972) and thus "more handy" than the full sets $\mathbf{N}_{p,q}$ or $\mathbf{N} \setminus \mathbf{N}_{p,q}$.

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.

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 3.10 (Observations and Difference). On a transition system $\mathcal{S}$, the possible observations of a state, $\llbracket \cdot \rrbracket^{👁} \colon \mathcal{P} \to 2^{\mathsf{HML}}$, are defined as:

$$\llbracket p \rrbracket^{👁} := \{\varphi \in \mathsf{HML} \mid p \in \llbracket \varphi \rrbracket\}.$$

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

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

$\Delta(p, q)$ 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 Definition 2.12:

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

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

A spectroscopy then is about computing some abstraction Δ_α such that $N \in \Delta_\alpha(p, q)$ precisely if $\Delta(p, q) \cap \mathcal{O}_N \neq \emptyset$.

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 (Balcázar et al., 1992; Hüttel & Shukla, 1996).

Bisimilarity finds itself in a *valley* of tractability, if we look at a cross section through the equivalence spectrum as in Figure 3.9. The best known bisimilarity algorithms for finite-state transition systems take $O(|{\to}| \log |S|)$ time (usually deriving from Paige & Tarjan, 1987).

For coarser simulation-like equivalences, the best known algorithms need $O(|{\to}|\,|S|)$ time (Ranzato & Tapparo, 2010).[61]

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

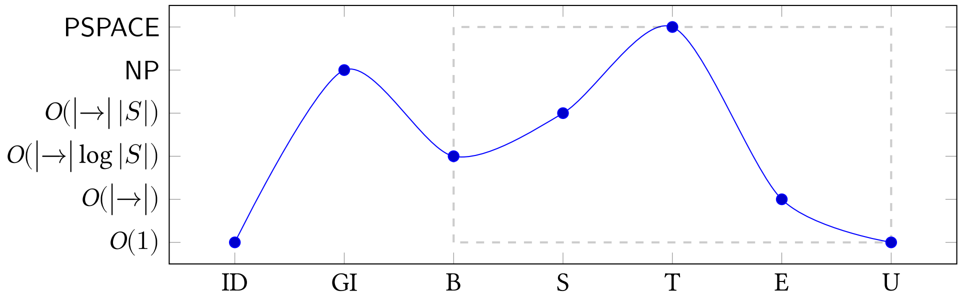


Figure 3.9: Bisimilarity's complexity valley.

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 however are little 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 $O(|{\rightarrow}|\log|S|)$. 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 3.9. 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* (Note 3), and instantiated the approach to the strong spectrum of Glabbeek (2001). From there, we have introduced the *spectroscopy problem*, which asks for notions to preorder compared states (Note 4).

By formulating the problem in terms of a lattice over $\mathbb{N}_\infty$-vectors, the family of qualitative strong preorder/equivalence problems becomes a single quantitative problem: The spectroscopy problem for the strong spectrum.

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

The core ideas of this section have already been explored in Bisping et al. (2022) and Bisping (2023a) 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 force observations in clauses of conjunctions and 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.

So far, we have a problem and confidence that its solution conveys information about equivalences of the strong spectrum. Subsection 3.3.3 has established that spectroscopy complexity must be at least PSPACE on the strong spectrum.

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

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. Secondly, conceptually—we just use the bisimulation game of Chapter 2, almost directly!

We already know 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:

> **Idea 5: 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 as defender's winning energy budgets by adding three-dimensional energies to the bisimulation game.

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

> **Idea 6: Computing cheapest wins**
>
> Attacker's winning budgets in energy reachability games can be computed by a generalized shortest paths algorithm.

Section 4.3 will provide 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*, which is a generalization of invertibility on monotonic functions.

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

Effectively, we adapt the game framework of Figure 2.13 to not treat one equivalence, but a spectrum of equivalences. This approach is summarized in Figure 4.1.

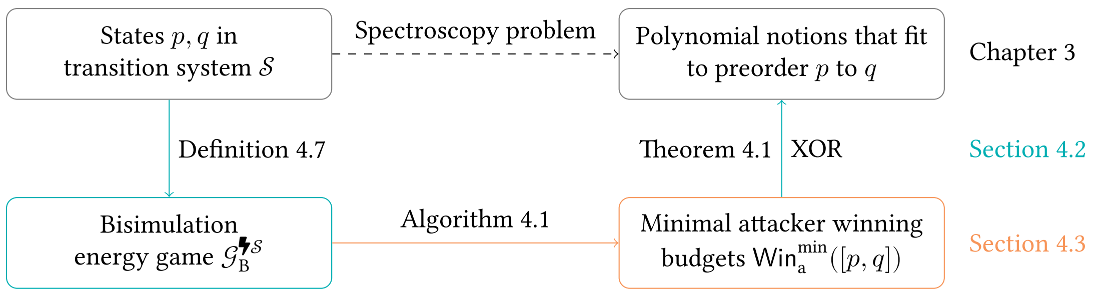

States p, q in transition system $\mathcal{S}$
Spectroscopy problem
Polynomial notions that fit to preorder p to q
Chapter 3
Definition 4.7
Theorem 4.1
XOR
Section 4.2
Bisimulation energy game $\mathcal{G}_{\mathrm{B}}^{\lightning\mathcal{S}}$
Algorithm 4.1
Minimal attacker winning budgets $\mathsf{Win}_{\mathrm{a}}^{\min}([p,q])$
Section 4.3

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

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* (Subsection 4.1.1) and then zoom in on *declining energy games* (Subsection 4.3), which include updates that combine vector components by taking their minimum. The latter are exactly what we need for the following Section 4.2.

4.1.1 Monotonic Energy Games

To introduce energy games, we generalize the definitions on games (Definition 2.15 and the following) to include *energy levels*. We define how winning regions then become 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* $\mathcal{G}^{\lightning}$ is a reachability game $(G, G_{\mathrm{d}}, \rightarrowtail)$ extended by an edge labeling of energy updates $\mathit{upd} \colon (\rightarrowtail) \to (\mathcal{E} \to (\{\bot\} \cup \mathcal{E}))$.

We demand the following for monotonic energy games:

- $\bot \notin \mathcal{E}$ is final in the sense that $\mathit{upd}(m)(\bot) = \bot$.
- All update functions $u = \mathit{upd}(m)$ with $m \in \rightarrowtail$ are monotonic and upward-closed with respect to $\leq$. More formally, this means that for

any energies $e, e' \in \mathcal{E}$, if $e \in \mathrm{dom}(u)$ and $e \leq e'$, then $e' \in \mathrm{dom}(u)$ and $u(e) \leq u(e')$, where $\mathrm{dom}(u) := \{e \in \mathcal{E} \mid u(e) \neq \bot\}$.

We denote by $\mathcal{G}^{\text{ϟ}}(g_0, e_0)$ the game played from starting position $g_0 \in G$ with initial energy level $e_0 \in \mathcal{E}$.

Definition 4.2 (Energy Level). For a finite play $\rho = g_0 g_1 ... g_{n-1} \in G^*$ of $\mathcal{G}^{\text{ϟ}}(g_0, e_0)$, its energy level $\mathrm{EL}(\rho)$ is computed recursively:

- $\mathrm{EL}(g_0) := e_0$
- $\mathrm{EL}(g_0 ... g_{i+1}) := \mathit{upd}(g_i, g_{i+1})(\mathrm{EL}(g_0 ... g_i))$

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

We consider the attacker to be energy-bounded and understand $\mathrm{EL}(\rho) = \bot$ to mean that they have run out of energy. Thus, we declare plays with $\mathrm{EL}(\rho) = \bot$ 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 2.17 to be *energy-positional* in the sense that they pick next moves depending on the current energy level, i.e. $f : G_\mathrm{a} \times \mathcal{E} \to G$.

Instead of a winning region as in Definition 2.18, we define *winning budgets* for game positions:

Definition 4.3 (Winning Budgets). For each position $g_0 \in G$ of energy game $\mathcal{G}^{\text{ϟ}}$, the *attacker winning energy budgets*, $\mathsf{Win}_\mathrm{a}(g_0) \subseteq \mathcal{E}$ are the energies e_0 where the attacker wins $\mathcal{G}^{\text{ϟ}}(g_0, e_0)$. The defender winning budgets Win_d are defined analogously.

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

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

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

Can the attacker win from g_1 with energy $(0, 0)$?

No, as all outgoing paths would lead to $\bot$-energy and thus to the defender winning.

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.

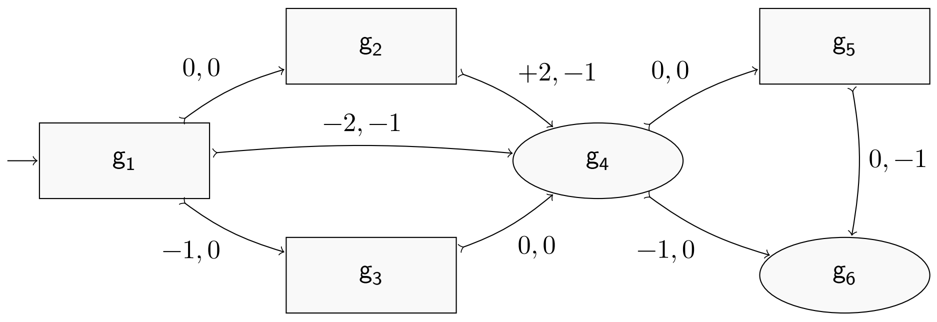


Figure 4.2: Simple energy game of Example 4.1.

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

Summing up our observations, $(0, 2)$ and $(2, 1)$ define a Pareto front of minimal budgets where the attacker wins from g_4, as depicted in Figure 4.3.

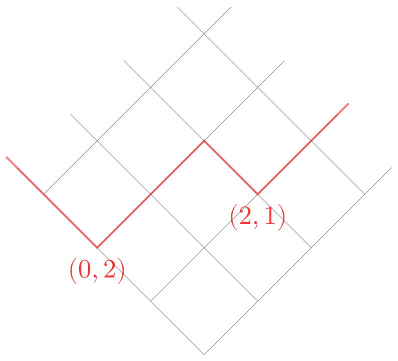


Figure 4.3: Attacker's Pareto front for g_1 of example energy game.

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 with 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.

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

Proposition 4.1. *Given an energy game $\mathcal{G}$, the attacker winning budgets $\mathsf{Win}_\mathrm{a}^\mathcal{G}$ are point-wise upward-closed, that is, $\uparrow \mathsf{Win}_\mathrm{a}^\mathcal{G}(g) = \mathsf{Win}_\mathrm{a}^\mathcal{G}(g)$ for all positions $g \in G$.*[62]

Proposition 4.2. *Given an energy game $\mathcal{G}$, the attacker winning budgets $\mathsf{Win}_\mathrm{a}^\mathcal{G}$ are characterized inductively by the following rules:*[63]

$$\frac{g_\mathrm{a} \in G_\mathrm{a} \quad g_\mathrm{a} \rightarrowtail g' \quad upd(g_\mathrm{a}, g')(e) \in \mathsf{Win}_\mathrm{a}^\mathcal{G}(g')}{e \in \mathsf{Win}_\mathrm{a}^\mathcal{G}(g_\mathrm{a})}$$

$$\frac{g_\mathrm{d} \in G_\mathrm{d} \quad \forall u, g'.\ g_\mathrm{d} \rightarrowtail g' \longrightarrow upd(g_\mathrm{d}, g')(e) \in \mathsf{Win}_\mathrm{a}^\mathcal{G}(g')}{e \in \mathsf{Win}_\mathrm{a}^\mathcal{G}(g_\mathrm{d})}$$

The inductive definition is nice as it allows for local proves of how the attacker wins. Defender's wins on the other hand would dually be a coinductive concept because the defender wins loops.[64]

[62] This is formalized as Lemma 3.2 in Lemke (2024).

[63] Proved as Theorem 3 in Lemke (2024).

[64] 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.

Remark 4.1. Ordinary reachability games can be seen as a special case of energy games with trivial energies, e.g. $\mathcal{E} = \{1\}$ and $upd(\cdot) = \mathrm{id}_{\mathcal{E}}$.

Remark 4.2. 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 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 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 special kind of monotonic energy games with vector-valued energy levels where updates will never increase energy levels in any component. An N-dimensional *declining energy game* is played on energy vectors:

Definition 4.4 (Energies and Energy Updates). For dimensionality N, the set of *energies*, **En**, is given by $(\mathbb{N} \cup \{\infty\})^N$.

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

- $u_k \in \{-1, 0\}$ (relative update), or
- $u_k = \mathtt{min}_D$ where $D \subseteq \{1, \ldots, N\}$ and $k \in D$ (minimum selection update).

Applying an update to an energy, $\mathsf{upd}(u, e)$, where $e = (e_1, \ldots, e_N) \in \mathbf{En}$ and $u = (u_1, \ldots, u_N) \in \mathbf{Up}$, yields a new energy vector e' where kth components $e'_k := e_k + u_k$ for $u_k \in \mathbb{Z}$ and $e'_k := \min_{d \in D} e_d$ for $u_k = \mathtt{min}_D$. Updates that would cause any component to become negative yield $\bot$.

Example 4.2. Consider the update label $(\mathtt{min}_{\{1,2\}}, 0, -1)$.

- $\mathsf{upd}((\mathtt{min}_{\{1,2\}}, 0, -1), (1, 1, 0))$ equals $\bot$ because of the last component.
- $\mathsf{upd}((\mathtt{min}_{\{1,2\}}, 0, -1), (2, 1, 1))$ equals $(1, 1, 0)$.

- $\mathsf{upd}((\mathtt{min}_{\{1,2\}}, 0, -1), (1, 1, 1))$ equals $(1, 1, 0)$, as well.
- There is no $e \in \mathbf{En}$ such that $\mathsf{upd}((\mathtt{min}_{\{1,2\}}, 0, -1), e) = (1, 0, 0)$, because the second component would demand a lower first component.

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

Definition 4.5 (Declining Energy Game). Given a weight labeling $w \colon (\rightarrowtail) \to \mathbf{Up}$, an N-dimensional *declining energy game* $\mathcal{G}^{\text{\Lightning}}$ is an energy reachability game with energies $\mathcal{E} := \mathbf{En}$ and with edges labeled $upd(m) := \mathsf{upd}(w(m), \cdot)$.

We write $g \overset{u}{\rightarrowtail} g'$ for a move $g \rightarrowtail g'$ 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.

Proposition 4.3. *In this model, energy levels can only decline.*

- *Updates may only decrease energies,* $\mathsf{upd}(u, e) \leq e$.
- *Updates are monotonic as demanded: If* $e_A \leq e_B$, *then* $\mathsf{upd}(u, e_A) \leq \mathsf{upd}(u, e_B)$.<

4.2 Characterizing the P-Time Part of the Spectrum

As hinted at in Subsection 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). In this section, we will show how to adapt the standard bisimulation game of Definition 2.20 to decide not just bisimulation, but *all polynomial-time* equivalences of the strong spectrum *at once.*

4.2.1 The Polynomial 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 polynomial slice, we only use three dimensions:

1. modal depth
2. depth of negative clauses, and
3. nesting depth of negations (that is, the first, fifth and sixth dimension of the strong spectrum Definition 3.7).

Components for other dimensions of the strong spectrum of the previous chapter are effectively set to ∞.

Definition 4.6 (Polynomial-Time Strong Spectrum). We denote as *polynomial-time strong spectrum* the projection of the strong spectrum (Definition 3.7) to the first, fifth, and sixth dimension, thus by notions

$$\mathbf{N}^{\mathsf{poly-strong}} := \mathbb{N}^3_\infty,$$

and the family of strong observation languages

$$\mathcal{O}^{\mathsf{poly-strong}}_{(N_1,N_2,N_3)\in\mathbf{N}^{\mathsf{poly-strong}}} := \mathcal{O}^{\mathsf{strong}}_{(N_1,\infty,\infty,\infty,N_2,N_3)}.$$

In effect, $\mathcal{O}_{N\in\mathbf{N}^{\mathsf{poly-strong}}}$ can be characterized by φ^N in the following grammar:

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

Figure 4.4 gives names to coordinates that correspond to notions discussed previously.

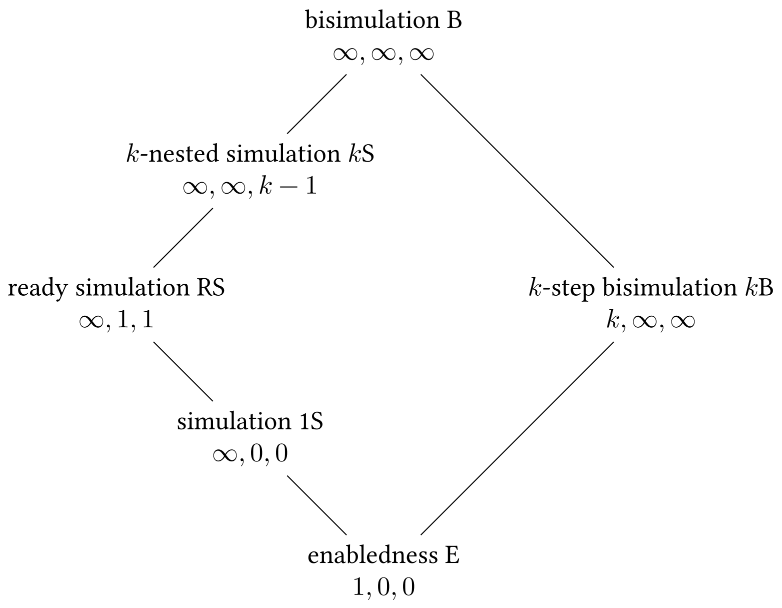


Figure 4.4: Hierarchy of polynomial-time decidable equivalences/preorders.

Except for enabledness, E, all coordinates after unfolding them for $\mathbf{N}^{\mathsf{strong}}$ are identical to the ones used previously. When one 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. $\bigwedge\{\langle \mathsf{a}\rangle, \langle \mathsf{b}\rangle\}$), but it is equally distinctive.[65]

[65] lemma Priced_Spectrum.lts.enabledness_conjunctions_are_neutral

4.2.2 The Bisimulation *Energy* Game

Let us upgrade the bisimulation game of Definition 2.20 with energies. The claim is that the resulting game $\mathcal{G}^{\lightning}_{\mathrm{B}}$ will characterize all equivalences of the polynomial-time strong spectrum in the sense that $p \preceq_{\mathcal{O}_e} q$ for $e \in \mathbf{N}^{\mathsf{poly-strong}}$ precisely if the defender wins $\mathcal{G}^{\lightning}_{\mathrm{B}}$ from $[p, q]$ with budget e.

As we have seen in Subsection 2.4.4, the game moves match the distinctive power of productions in the $\mathcal{O}_{\lfloor \mathrm{B} \rfloor}$-grammar of Example 2.11. 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 Subsection 4.2.1.[66]

[66] 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.

We add the computations that happen in the grammar of Definition 4.6 as energy updates to the bisimulation game and obtain:

Definition 4.7 (Bisimulation Energy Game). For a transition system $\mathcal{S}$, the *bisimulation energy game* $\mathcal{G}^{\lightning\mathcal{S}}_{\mathrm{B}}$ is played on the same graph as the *bisimulation game* $\mathcal{G}^{\mathcal{S}}_{\mathrm{B}}$ (of Definition 2.20), but the moves are weighted by the following energy updates:

- Attacker swaps are counted as a negation and limit the further observations to the depth of negative clauses:
$$[p, q] \quad \xrightarrow{\mathtt{min}_{\{1,2\}}, 0, -1}_{\mathrm{B}}\!\!{}^{\lightning} \quad [q, p].$$
- Simulation challenges count as an observation (using up the budget for modal depth):
$$[p, q] \quad \xrightarrow{-1, 0, 0}_{\mathrm{B}}\!\!{}^{\lightning} \quad (\alpha, p', q) \quad \text{if} \quad p \xrightarrow{\alpha} p'.$$
- And defender answers come for free:
$$(\alpha, p', q) \quad \xrightarrow{0, 0, 0}_{\mathrm{B}}\!\!{}^{\lightning} \quad [p', q'] \quad \text{if} \quad q \xrightarrow{\alpha} q'.$$

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 4.5 shows the game graph, also exhibiting the cheapest formulas with regards to the polynomial spectrum that correspond to attacker winning strategies.

The attacker wins $\mathcal{G}^{\lightning}_{\mathrm{B}}$ from $[\mathsf{Q}, \mathsf{T}]$ if they start out with an energy budget of $(2, 2, 2)$ or above. 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 second bound implies that the budgets, $(\infty, 1, 1)$ and $(\infty, 0, 0)$ are won by the defender as well.

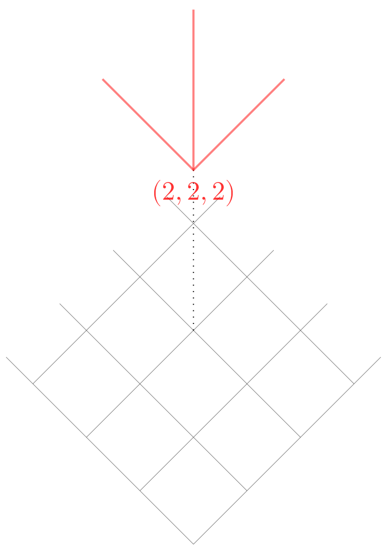


Figure 4.6: Attacker's Pareto front for $[\mathsf{Q}, \mathsf{T}]$ of the example bisim energy game.

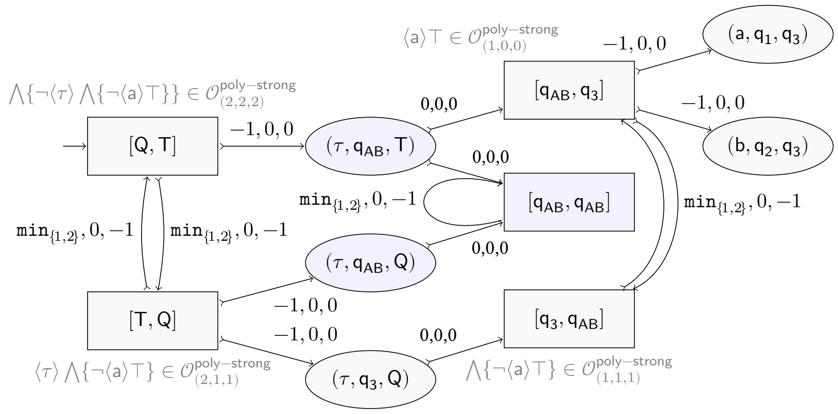

$\langle \mathsf{a} \rangle \top \in \mathcal{O}^{\mathsf{poly-strong}}_{(1,0,0)}$
$-1, 0, 0$
$(\mathsf{a}, \mathsf{q}_1, \mathsf{q}_3)$
$\bigwedge\{\neg\langle\tau\rangle \bigwedge\{\neg\langle \mathsf{a}\rangle\top\}\} \in \mathcal{O}^{\mathsf{poly-strong}}_{(2,2,2)}$
$0,0,0$
$[\mathsf{q}_{\mathsf{AB}}, \mathsf{q}_3]$
$-1, 0, 0$
$(\mathsf{b}, \mathsf{q}_2, \mathsf{q}_3)$
$[\mathsf{Q}, \mathsf{T}]$
$-1, 0, 0$
$(\tau, \mathsf{q}_{\mathsf{AB}}, \mathsf{T})$
$0,0,0$
$\mathtt{min}_{\{1,2\}}, 0, -1$
$[\mathsf{q}_{\mathsf{AB}}, \mathsf{q}_{\mathsf{AB}}]$
$\mathtt{min}_{\{1,2\}}, 0, -1$
$\mathtt{min}_{\{1,2\}}, 0, -1$
$\mathtt{min}_{\{1,2\}}, 0, -1$
$(\tau, \mathsf{q}_{\mathsf{AB}}, \mathsf{Q})$
$0,0,0$
$-1, 0, 0$
$[\mathsf{T}, \mathsf{Q}]$
$-1, 0, 0$
$0,0,0$
$[\mathsf{q}_3, \mathsf{q}_{\mathsf{AB}}]$
$\langle\tau\rangle \bigwedge\{\neg\langle \mathsf{a}\rangle\top\} \in \mathcal{O}^{\mathsf{poly-strong}}_{(2,1,1)}$
$(\tau, \mathsf{q}_3, \mathsf{Q})$
$\bigwedge\{\neg\langle \mathsf{a}\rangle\top\} \in \mathcal{O}^{\mathsf{poly-strong}}_{(1,1,1)}$

Figure 4.5: The bisimulation energy game of Example 4.3 with energy updates and implied formulas.

On the game-side, this tells us is 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 $\mathcal{O}_{\lfloor \mathrm{B} \rfloor}$.

Thus, $\mathsf{N}^{\mathsf{poly-strong}}_{\mathsf{Q},\mathsf{T}} = \{N \in \mathsf{N}^{\mathsf{poly-strong}} \mid N \not\leq (2,2,2)\}$ is the solution for the (polynomial-time 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 3.6.

4.2.3 Correctness of Characterization

We now prove that the bisimulation energy game indeed characterizes the equivalences of the polynomial-time strong spectrum. In following chapters, we will prove stronger results for more general games in more detail. The approach is to generalize the connection between winning attacks and distinguishing formulas that we have already explored in Lemma 2.10 and Lemma 2.11 to energies and spectrum.

Definition 4.8 (Strategy Formulas for $\mathcal{G}^{\lightning}_{\mathrm{B}}$). The set of *strategy formulas* for a game position g and a budget e, $\mathsf{Strat}_{\mathrm{B}}(g, e)$, in the context of a bisimulation energy game $\mathcal{G}_{\mathrm{B}}^{\lightning\mathcal{S}}$ is defined inductively by the following rules:

$$\frac{[p,q] \overset{-1,0,0}{\rightarrowtail}{}^{\lightning}_{\mathrm{B}} (\alpha,p',q) \quad e' = \mathsf{upd}((-1,0,0),e) \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}((\alpha,p',q)) \quad \varphi \in \mathsf{Strat}_{\mathrm{B}}((\alpha,p',q),e')}{(\langle\alpha\rangle\varphi) \in \mathsf{Strat}_{\mathrm{B}}([p,q],e)}$$

$$\frac{\forall q' \in \mathrm{Der}(q,\alpha). \quad (\alpha,p',q) \overset{0,0,0}{\rightarrowtail}{}^{\lightning}_{\mathrm{B}} [p',q'] \ \wedge\ e \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}([p',q']) \ \wedge\ \psi_{q'} \in \mathsf{Strat}_{\mathrm{B}}([p',q'],e)}{(\bigwedge_{q' \in \mathrm{Der}(q,\alpha)} \psi_{q'}) \in \mathsf{Strat}_{\mathrm{B}}((\alpha,p',q),e)}$$

$$\frac{[p,q] \overset{\mathtt{min}_{\{1,2\}},0,-1}{\rightarrowtail}{}^{\lightning}_{\mathrm{B}} [q,p] \quad e' = \mathsf{upd}((\mathtt{min}_{\{1,2\}},0,-1),e) \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}([q,p]) \quad \langle\alpha\rangle\varphi \in \mathsf{Strat}_{\mathrm{B}}([q,p],e')}{(\neg\langle\alpha\rangle\varphi) \in \mathsf{Strat}_{\mathrm{B}}([p,q],e)}$$

Effectively, this definition generalizes the construction of distinguishing formulas on $\mathcal{G}_{\mathrm{B}}$ we introduced in Lemma 2.10. So the formulas we encountered before also serve as examples of such strategy formulas.

Lemma 4.1. *If* $e \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}([p,q])$, *then there is* $\varphi \in \mathsf{Strat}_{\mathrm{B}}([p,q],e)$ *with* $\varphi \in \mathcal{O}_e^{\mathsf{poly-strong}}$, $p \in [\![\varphi]\!]$ *and* $q \notin [\![\varphi]\!]$.

Proof. We prove the following, more general, property by induction.

1. If $e \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}([p,q])$, then there is $\psi \in \mathsf{Strat}_{\mathrm{B}}([p,q],e)$ of form $\psi = \langle\alpha\rangle\varphi'$ or $\neg\langle\alpha\rangle\varphi'$ with $\bigwedge\{\psi\} \in \mathcal{O}_e^{\mathsf{poly-strong}}$, $p \in [\![\psi]\!]$ and $q \notin [\![\psi]\!]$.
2. If $e \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}((\alpha,p',q))$, then there is $\varphi \in \mathsf{Strat}_{\mathrm{B}}((\alpha,p',q),e)$ of form $\varphi = \bigwedge\Psi$ with $\varphi \in \mathcal{O}_e^{\mathsf{poly-strong}}$, $p' \in [\![\varphi]\!]$ and $q \notin [\![\langle\alpha\rangle\varphi]\!]$.

Proof by induction on the inductive characterizations of attacker winning budgets Proposition 4.2.

1. Assume $e \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}([p,q])$. This must be due to one of the following moves:
   - Case $[p,q] \rightarrowtail^{\lightning}_{\mathrm{B}} [q,p]$ with $e' = \mathsf{upd}((\mathtt{min}_{\{1,2\}},0,-1),e) \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}([q,p])$. By induction hypothesis on $[q,p]$, there is $\psi \in \mathsf{Strat}_{\mathrm{B}}([q,p],e')$ with $\bigwedge\{\psi\} \in \mathcal{O}_{e'}^{\mathsf{poly-strong}}$, $q \in [\![\psi]\!]$ and $p \notin [\![\psi]\!]$. Consider the possible forms of ψ:
     – $\psi = \langle\alpha\rangle\varphi'$. By Definition 4.8 of $\mathsf{Strat}_{\mathrm{B}}$, we obtain $\neg\langle\alpha\rangle\varphi' \in \mathsf{Strat}_{\mathrm{B}}([p,q],e)$. Because of the semantics of HML, $\neg\langle\alpha\rangle\varphi'$ must distinguish p from q. Also, $\bigwedge\{\neg\langle\alpha\rangle\varphi'\} \in \mathcal{O}_e^{\mathsf{poly-strong}}$, by the calculations of the grammar in Definition 4.6.
     – $\psi = \neg\langle\alpha\rangle\varphi'$. This can only be in $\mathsf{Strat}_{\mathrm{B}}([q,p],e')$ due to φ' being in $\mathsf{Strat}_{\mathrm{B}}([p,q],\mathsf{upd}((\mathtt{min}_{\{1,2\}},0,-1),e'))$. Clearly, φ' must distinguish p from q and $\bigwedge\{\varphi'\} \in \mathcal{O}_e^{\mathsf{poly-strong}}$.
   - Case $[p,q] \rightarrowtail^{\lightning}_{\mathrm{B}} (\alpha,p',q)$ with $p \xrightarrow{\alpha} p'$ and $e' = \mathsf{upd}((-1,0,0),e) \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}((\alpha,p',q))$. By induction hypothesis on (α,p',q), there is $\bigwedge\Psi \in \mathsf{Strat}_{\mathrm{B}}((\alpha,p',q),e')$ and $\bigwedge\Psi \in \mathcal{O}_{e'}^{\mathsf{poly-strong}}$ with

$p' \in [\![\bigwedge \Psi]\!]$ and $q \notin [\![\langle\alpha\rangle \bigwedge \Psi]\!]$. By Definition 4.8, we obtain $\langle\alpha\rangle \bigwedge \Psi \in \mathsf{Strat}_{\mathrm{B}}([p,q],e)$. Due to the semantics of HML and $p \xrightarrow{\alpha} p'$, $p \in [\![\langle\alpha\rangle \bigwedge \Psi]\!]$, thus distinguishing p from q. Also, $\langle\alpha\rangle \bigwedge \Psi \in \mathcal{O}_e^{\mathsf{poly-strong}}$ as $\bigwedge \Psi \in \mathcal{O}_{e-\hat{\mathbf{e}}_1}^{\mathsf{poly-strong}}$.

2. Assume $e \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}((\alpha, p', q))$. This means that e suffices for the attacker to win every q'-move $(\alpha, p', q) \rightarrowtail_{\mathrm{B}}^{\lightning} [p', q']$ with $q \xrightarrow{\alpha} q'$ that the defender might take, with $e \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}([p', q'])$. For each q', we employ the induction hypothesis to obtain $\psi_{q'} \in \mathsf{Strat}_{\mathrm{B}}([p', q'], e)$ of form $\psi_{q'} = \langle\alpha\rangle\varphi'$ or $\neg\langle\alpha\rangle\varphi'$ with $\bigwedge\{\psi_{q'}\} \in \mathcal{O}_e^{\mathsf{poly-strong}}$, $p' \in [\![\psi_{q'}]\!]$ and $q' \notin [\![\psi_{q'}]\!]$. By Definition 4.8, $(\bigwedge_{q' \in \mathrm{Der}(q,\alpha)} \psi_{q'}) \in \mathsf{Strat}_{\mathrm{B}}((\alpha, p', q), e)$. By the HML-semantics, $(\bigwedge_{q' \in \mathrm{Der}(q,\alpha)} \psi_{q'})$ must be true for p', and also taking the semantics of observation, $\langle\alpha\rangle(\bigwedge_{q' \in \mathrm{Der}(q,\alpha)} \psi_{q'})$ false for q. Moreover, $(\bigwedge_{q' \in \mathrm{Der}(q,\alpha)} \psi_{q'}) \in \mathcal{O}_e^{\mathsf{poly-strong}}$ by the grammar. □

Lemma 4.2. *If there is $\varphi \in \mathcal{O}_e^{\mathsf{poly-strong}}$ with $p \in [\![\varphi]\!]$ and $q \notin [\![\varphi]\!]$, then $e \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}([p,q])$.*

Proof. By induction on the grammar of $\mathcal{O}_e^{\mathsf{poly-strong}}$.

Consider the cases of $\varphi \in \mathcal{O}_e^{\mathsf{poly-strong}}$ with $p \in [\![\varphi]\!]$ and $q \notin [\![\varphi]\!]$.

- $\varphi = \top$. This cannot be the case as $q \notin [\![\varphi]\!]$ and $[\![\top]\!] = \mathcal{P}$.
- $\varphi = \langle\alpha\rangle\varphi'$ with $\varphi' \in \mathcal{O}_{e-\hat{\mathbf{e}}_1}^{\mathsf{poly-strong}}$. As φ distinguishes p from q, there must be a $p' \in [\![\varphi']\!]$ such that $p \xrightarrow{\alpha} p'$ and, for all $q' \in \mathrm{Der}(q, \alpha)$, $q' \notin [\![\varphi']\!]$. Due to the induction hypothesis, $e - \hat{\mathbf{e}}_1 \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}([p', q'])$ for all such $q' \in \mathrm{Der}(q, \alpha)$. Therefore, $e - \hat{\mathbf{e}}_1 \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}((\alpha, p', q))$. As the attacker can move $[p,q] \xrightarrow{-\hat{\mathbf{e}}_1} (\alpha, p', q)$, this proves $e \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}([p,q])$ by Proposition 4.2.
- $\varphi = \bigwedge\{\Psi\}$ one of the $\psi \in \Psi$ must be false for q. Consider the possible forms of ψ:
  - Case $\psi = \langle\alpha\rangle\varphi'$ with $\varphi' \in \mathcal{O}_{e-\hat{\mathbf{e}}_1}^{\mathsf{poly-strong}}$. Then we can apply the same argument as in the case of $\varphi = \langle\alpha\rangle\varphi'$.
  - Case $\psi = \neg\langle\alpha\rangle\varphi'$ with $\varphi' \in \mathcal{O}_{(e \sqcap (e_2, \infty, \infty)) - \hat{\mathbf{e}}_1 - \hat{\mathbf{e}}_3}^{\mathsf{poly-strong}}$. Therefore, $\langle\alpha\rangle\varphi' \in \mathcal{O}_{(e \sqcap (e_2, \infty, \infty)) - \hat{\mathbf{e}}_3}^{\mathsf{poly-strong}}$ 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 $(e \sqcap (e_2, \infty, \infty)) - \hat{\mathbf{e}}_3 \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}([q,p])$. The move $[p,q] \xrightarrow{\mathtt{min}_{\{1,2\}}, 0, -1}_{\mathrm{B}}^{\lightning} [q,p]$ and $\mathsf{upd}((\mathtt{min}_{\{1,2\}}, 0, -1), e) = (e \sqcap (e_2, \infty, \infty)) - \hat{\mathbf{e}}_3$ justify that $e \in \mathsf{Win}_{\mathrm{a}}^{\mathcal{G}_{\mathrm{B}}^{\lightning}}([p,q])$. □

Theorem 4.1. *$p \preceq_{\mathcal{O}_e} q$ for $e \in \mathsf{N}^{\mathsf{poly-strong}}$ precisely if the defender wins $\mathcal{G}_{\mathrm{B}}^{\lightning}$ from $[p,q]$ with budget e.*

Thus, we have established that the bisimulation energy game $\mathcal{G}_{\mathrm{B}}^{\lightning}$ characterizes the equivalences of the polynomial-time strong spectrum $\mathsf{N}^{\mathsf{poly-strong}}$. 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 which energy budgets are winning for the attacker (or the defender) at positions of an energy game:

Definition 4.9 (Monotonic Energy Game Winner Problem). In the context of an energy system $(\mathcal{E}, \leq)$ and a finite energy game $\mathcal{G}^{\lightning} = (G, G_{\mathrm{d}}, \rightarrowtail, upd)$, the *energy game winner problem* goes:

Input Game position $g \in G$.
Output $\mathsf{Win}_{\mathrm{a}}^{\mathrm{min}}(g)$—the Pareto front of minimal winning budgets for the attacker at g.

We will show how to solve the problem for energy games with monotonic updates. 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 see how to 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, and this is what the variable naming in the following definition alludes to.

Definition 4.10. A *Galois connection* between two partially ordered sets $(C, \leq_C)$ and $(A, \leq_A)$ is a pair of functions $\alpha\colon C \to A$ and $\gamma\colon A \to C$ 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 4.7. (The illustration also illuminates why the α-function is often referred to as "lower adjoint" and γ as "upper adjoint" of a Galois connection.)

Example 4.4. Take the non-negative reals $\mathbb{R}^{\geq 0}$ as concrete domain and the natural numbers $\mathbb{N}$ as abstract domain. Then, as function $\alpha_{\mathbb{R}}\colon \mathbb{R}^{\geq 0} \to \mathbb{N}$,

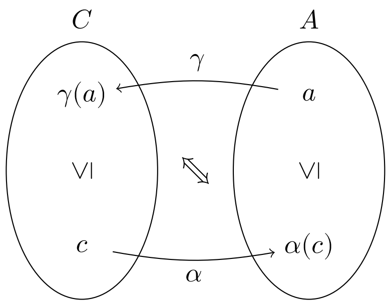


Figure 4.7: Visualization of Galois connections according to Definition 4.10.

take flooring, $x \mapsto \lfloor x \rfloor$, and for the other direction, the identity $\mathrm{id}_{\mathbb{N}}$, $n \mapsto n$. Clearly, $\lfloor x \rfloor \leq n \iff x \leq n$. Therefore, $\alpha_{\mathbb{R}}$ and $\mathrm{id}_{\mathbb{N}}$ form a Galois connection.

Any monotonic function naturally induces a Galois-connected abstraction function.

Lemma 4.3. *The following are equivalent:*

- *$\gamma\colon A \to C$ is a monotonic function, and $\alpha(c) = \inf\{a \mid c \leq_C \gamma(a)\}$ for all $c \in C$.*
- *α and γ constitute a Galois connection between C and A.*[67]

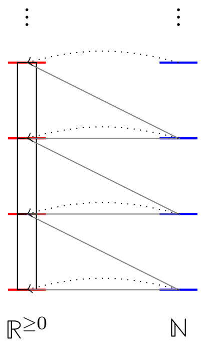


Figure 4.8: Illustration of the Galois connection of Example 4.4 between $\mathbb{R}^{\geq 0}$ and $\mathbb{N}$ using $\lfloor \cdot \rfloor$-function, whose mapping is illustrated by the gray triangles.

[67] Proposition 4 of Erné et al. (1993).

We will refer to functions derived like $\alpha(c)$ in Lemma 4.3 as "undo" functions. We understand them to generalize inverse functions of monotonic functions as these will always be Galois connected.

Lemma 4.4. *If a function $f\colon A \to C$ is surjective and injective, it has a unique inverse function $f^{-1}\colon C \to A$ such that $f^{-1} \circ f = \mathrm{id}_A$ and $f \circ f^{-1} = \mathrm{id}_C$.*

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

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

and

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

□

In the following, we will use a $\circlearrowleft$-symbol to label the adjoint functions that we use to undo energy updates.

Remark 4.3. Definition 3.5 demands monotonicity of our observation languages for a spectrum, $\mathcal{O}_{N \in \mathbf{N}}$. 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:

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

The singleton case of this function exactly matches our expressiveness prices of Definition 3.6:

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

4.3.2 The Algorithm

To implement a back-propagation algorithm, we have to assume that we know a function that undoes energy updates, $\mathit{upd}^{\circlearrowleft}\colon (\rightarrowtail) \to \mathcal{E} \to \mathcal{E}$ with

$\mathit{upd}^{\circlearrowleft}(g, g')(e') = \inf\{e \mid e' \leq \mathit{upd}(g, g')(e)\}$. That means, we assume that there is a Galois connection $(\mathit{upd}^{\circlearrowleft}, \mathit{upd})$ between energies $\mathcal{E}$ and the domain of updates $\{e \in \mathcal{E} \mid \mathit{upd}(e) \neq \bot\}$.

Algorithm 4.1 can be used to compute the minimal attacker winning budgets.

```
 1  def compute_winning_budgets($\mathcal{G}^{\lightning} = (G, G_\mathrm{d}, \rightarrowtail, \mathit{upd}), \mathit{upd}^{\circlearrowleft}$):
 2    attacker_win := $[g \mapsto \emptyset \mid g \in G]$
 3    todo := $\{g \in G_\mathrm{d} \mid g \not\rightarrowtail\}$
 4    while todo $\neq \emptyset$:
 5      g := some todo
 6      todo := todo $\setminus$ {g}
 7      if g $\in G_\mathrm{a}$:
 8        new_attacker_win := Min($\{\mathit{upd}^{\circlearrowleft}(\mathrm{g}, g')(e')$
            $\mid$ g $\rightarrowtail g' \wedge e' \in$ attacker_win$[g']\}$)
 9      else:
10        new_attacker_win := $\{\mathbf{0}\}$
11        for g' $\in$ (g $\rightarrowtail \cdot$):
12          new_attacker_win := Min($\{\sup(e_\mathrm{a}, \mathit{upd}^{\circlearrowleft}(\mathrm{g}, g')(e')) \mid$
              $e_\mathrm{a} \in$ new_attacker_win $\wedge$ g $\rightarrowtail g' \wedge e' \in$ attacker_win$[g']\}$)
13      if new_attacker_win $\neq$ attacker_win[g]:
14        todo := todo $\cup$ ($\cdot \rightarrowtail$ g)
15    $\mathsf{Win}_\mathrm{a}^\mathrm{min}$ := attacker_win
16    return $\mathsf{Win}_\mathrm{a}^\mathrm{min}$
```

Algorithm 4.1: Algorithm determining the minimal attacker winning budgets $\mathsf{Win}_\mathrm{a}^\mathrm{min}$ of an energy game $\mathcal{G}^{\lightning}$.

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). By $\mathbf{0}$, we denote the minimal element of the energy system $\mathcal{E}$.

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

Example 4.5. Let us execute Algorithm 4.1 on the $\mathbb{N}^2$-game of Example 4.1. The game graph, labeled by the minimal attacker winning energies we find, is reproduced in Figure 4.9.
What we need for the algorithm to run, is some undo function $\mathit{upd}^{\circlearrowleft} \colon (\rightarrowtail) \to \mathcal{E} \to \mathcal{E}$. We define it as:

$$\mathit{upd}^{\circlearrowleft}(e') := \sup(e - \vec{u}, \mathbf{0})$$

We list the steps of the computation in Example 4.5, and the mechanics behind

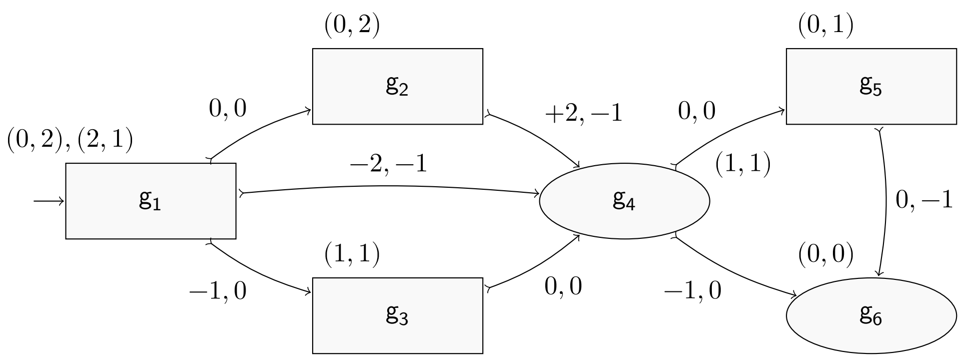


Figure 4.9: Computed winning budgets $\mathrm{Win}_a^{\min}$ on the game in Example 4.5.

them below.

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	$\{(0,2),(2,1)\}$	$\emptyset$

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 it its minimal attacker-winning budget to $\mathbf{0}$. 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 one successor `attacker_win`$[g_5] = \emptyset$.
3. $g_4 \in G_a$ is updated with the $upd^{\circlearrowleft}((0,-1))((0,0)) = \sup((0,0) - (0,-1), \mathbf{0}) = (0,1)$. 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 4.10. The 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. So we determine as a preliminary winning budget $upd^{\circlearrowleft}((-2,-1))((1,1)) = (3,2)$.
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 4.11 as $\mathrm{Min}\{(0,2),(2,1),(3,4)\} = \{(0,2),(2,1)\}$. This operation corresponds to taking the union of the

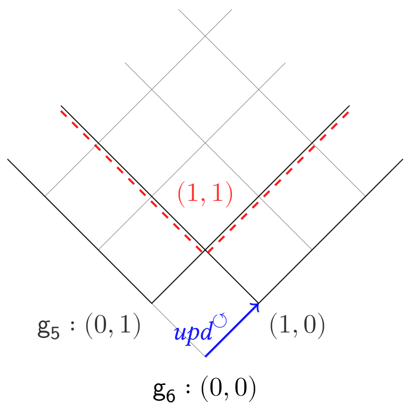


Figure 4.10: Multiplicative combination (dashed red) of successor Pareto fronts (black) at defender position g_4 in step 4.

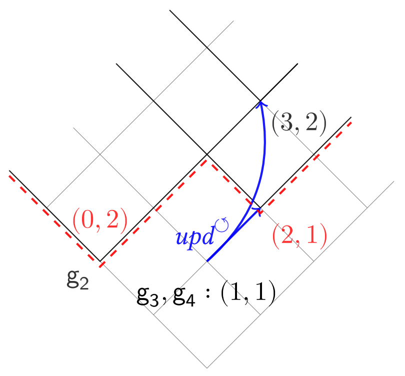


Figure 4.11: Additive combination (dashed red) of successor Pareto fronts (black) at attacker position g_1 in step 8.

$(3, 4)$ of step 5 is thus superseded. As `todo` is empty, the algorithm terminates.

Theorem 4.2. *Assume energies form a well-founded bounded $\sqcup$-semi-lattice $(\mathcal{E}, \leq)$. Given a finite-state energy game $\mathcal{G}^{\lightning} = (G, G_{\mathrm{d}}, \rightarrowtail, upd)$ with monotonic $upd^{\circlearrowleft}$-functions for all moves and with computable undo function $upd^{\circlearrowleft}(g, g')(e') = \min\{e \mid e' \leq upd(g, g')(e)\}$, Algorithm 4.1 computes the Pareto front of minimal attacker winning budgets, solving the* monotonic energy game winner problem *(Definition 4.9).*

Proof. Algorithm 4.1 is a work-list variant of a fixed-point iteration computing the least fixed point of attacker winning budgets $\mathsf{Win}_{\mathrm{a}}$ according to the inductive characterization of Proposition 4.2.

The "least" here refers to the size of the $\mathsf{Win}_{\mathrm{a}}$-sets characterized by the $\mathsf{Win}_{\mathrm{a}}^{\min}$-Pareto front.

A detailed proof, employing Kleene's fixed point theorem, can be found in 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. □

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*, $\mathrm{wdh}_B(b)$, 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.)

The *height*, $\mathrm{hgt}_B(b)$, below b is defined as the size of the maximal size of chains $B' \subseteq \downarrow\{b\}$.

Theorem 4.3. *Consider the following parameters of an energy game problem on $\mathcal{G}^{\lightning} = (G, G_{\mathrm{d}}, \rightarrowtail, upd)$ where the energies at least form a well-founded bounded $\sqcup$-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 $\mathcal{E}$,*
- *$t_{upd^{\circlearrowleft}}$, time to compute $upd^{\circlearrowleft}$ on an energy,*
- *$e_{\mathcal{G}}$, the highest energy that can be achieved by applying permutations of $upd^{\circlearrowleft}(\cdot)$ to $\mathbf{0}$ for $|G| - 1$ times.*
- *$o_{\rightarrowtail}$, the out-degree of the game graph $\rightarrowtail$.*

Then, Algorithm 4.1 computes the minimal attacker winning budgets in

$$O((|G| + \mathrm{hgt}_{\mathcal{E}}(e_{\mathcal{G}})) \cdot |G| \cdot o_{\rightarrowtail} \cdot (\mathrm{wdh}_{\mathcal{E}}(e_{\mathcal{G}}))^2 \cdot (t_{upd^{\circlearrowleft}} + t_{\mathrm{sup}} + \mathrm{wdh}_{\mathcal{E}}(e_{\mathcal{G}}) \cdot t_{\leq})).$$

If $upd^{\circlearrowleft}$ is strictly declining, then the following suffices:

$$O(|G|^2 \cdot o_{\rightarrowtail} \cdot (\mathrm{wdh}_{\mathcal{E}}(e_{\mathcal{G}}))^2 \cdot (t_{upd^{\circlearrowleft}} + t_{\sup} + \mathrm{wdh}_{\mathcal{E}}(e_{\mathcal{G}}) \cdot t_{\leq})).$$

In both cases, the algorithm needs $O(|G| \cdot \mathrm{wdh}_{\mathcal{E}}(e_{\mathcal{G}}) \cdot N)$ space to store the Pareto fronts.

Proof. As in Theorem 4.2, we refer to Lemke (2024) for a very detailed proof, where the corresponding fact is proved as Theorem 7.

TODO: Add a proof sketch regardless. □

4.3.4 Solving Declining Energy Games

Let us instantiate the previous two subsections to declining energy games. All we have to do is instantiate $upd^{\circlearrowleft}$ to obtain an algorithm and think about the structure of the energies to tell its complexity.

Definition 4.12. The undo-update function $\mathsf{upd}^{\circlearrowleft}$ on an N-dimensional declining energy game is defined component-wise for $k \in \{1...N\}$ as:

$$\begin{aligned}(\mathsf{upd}^{\circlearrowleft}(u, e'))_k := \max(\quad & \{e'_k - u_k \mid u_k \in \{0, -1\}\} \\ \cup \quad & \{e'_j \mid \exists D.\, u_j = \mathtt{min}_D \wedge k \in D\}).\end{aligned}$$

TODO: Example

Lemma 4.5. *For all updates $u \in \mathbf{Up}$, $\mathsf{upd}^{\circlearrowleft}(u, \cdot)$ and $\mathsf{upd}(u, \cdot)$ form a Galois connection on the domain* $\mathrm{dom}(u)$.

Proof. As $\mathsf{upd}(u, \cdot)$ is monotonic, all that needs to be shown according to Lemma 4.3, is that $\mathsf{upd}^{\circlearrowleft}(u, e') = \min\{e \mid e' \leq \mathsf{upd}(u, e)\}$.

- Soundness: $\mathsf{upd}^{\circlearrowleft}(u, e') \in \{e \mid e' \leq \mathsf{upd}(u, e)\}$. This boils down to showing $e'_k \leq (\mathsf{upd}(u, \mathsf{upd}^{\circlearrowleft}(u, e')))_k$. Consider the cases of u_k.
  - Case $u_k \in \{0, -1\}$. Then, we have to show $e'_k \leq \mathsf{upd}^{\circlearrowleft}(u, e') + u_k$. Due to $\mathsf{upd}^{\circlearrowleft}$ being increasing, this must be the case as $e'_k \leq e'_k + u_k$.
  - Case there is D such that $u_k = \mathtt{min}_D$ and $k \in D$. Then, by definition of $\mathtt{min}$-updates, we have to show $e'_k \leq \min\{(\mathsf{upd}^{\circlearrowleft}(u, e'))_j \mid j \in D\}$, that is, $e'_k \leq \mathsf{upd}^{\circlearrowleft}(u, e')_j$ for all $j \in D$. We do so by contradiction: Assume there were a j where this ordering does not hold, meaning $e'_k > \mathsf{upd}^{\circlearrowleft}(u, e')_j$. As $k \in D$, this is impossible due to the definition of $\mathsf{upd}^{\circlearrowleft}$ in terms of max.
- Minimality: For all $e' \leq \mathsf{upd}(u, e)$, we show $\mathsf{upd}^{\circlearrowleft}(u, e') \leq e$. More specifically, $(\mathsf{upd}^{\circlearrowleft}(u, e'))_k \leq e_k$. Let us define $d' := \max\{e'_j \mid \exists D.\, u_j = \mathtt{min}_D \wedge k \in D\}$. Consider the cases of $u_k \in \mathbf{Up}$.

– Case $u_k \in \{0, -1\}$ and $e'_k - u_k \geq d'$. Then $e'_k \leq (\mathsf{upd}(u, e))_k = e_k + u_k$, implying $e'_k - u_k \leq (\mathsf{upd}(u, e))_k = e_k$. By definition, $(\mathsf{upd}^{\circlearrowleft}(u, e'))_k = \max(e'_k - u_k, d')$. As $e'_k - u_k \geq d'$, we can connect to the previous inequality, getting $(\mathsf{upd}^{\circlearrowleft}(u, e'))_k = e'_k - u_k \leq e_k$.
– Otherwise, $(\mathsf{upd}^{\circlearrowleft}(u, e'))_k = d'$. We show $d' \leq e_k$ by contradiction. Assume it were the case that $d' > e_k$. For this to be true there would need to be j and D such that $u_j = \mathtt{min}_D$, $k \in D$, and $e'_j \geq d' > e_k$. By definition of `min`-update at j, we calculate $(\mathsf{upd}(u, e))_j \leq e_k < e'_j$. But this would contradict the global assumption that $e' \leq \mathsf{upd}(u, e)$. ☐

Lemma 4.6. *Algorithm 4.1 solves the energy game problem on an N-dimensional* declining *energy game $\mathcal{G}^{\lightning} = (G, G_{\mathrm{d}}, \rightarrowtail, w)$ in time*

$$O(o_{\rightarrowtail} \cdot |G|^{2 \cdot N} \cdot (N^2 + |G|^{N-1} \cdot N))$$

and space $O(|G|^N \cdot N)$.

Proof. For declining updates, we instantiate the second case of Theorem 4.3:

$$O(|G|^2 \cdot o_{\rightarrowtail} \cdot (\mathrm{wdh}_{\mathcal{E}}(e_{\mathcal{G}}))^2 \cdot (t_{upd^{\circlearrowleft}} + t_{\mathrm{sup}} + \mathrm{wdh}_{\mathcal{E}}(e_{\mathcal{G}}) \cdot t_{\leq})).$$

For *declining energy games* as defined in Definition 4.5, we can fill in several blanks:

- t_{sup}, the time to calculate the supremum between two elements of **En** is in $O(N)$,
- $t_{\leq}$, time to compare two elements of **En** is in $O(N)$,
- $t_{upd^{\circlearrowleft}}$, time to compute $\mathsf{upd}^{\circlearrowleft}$ on an energy is in $O(N^2)$,
- $e_{\mathcal{G}}$, the highest energy that can be achieved by applying permutations of $\mathsf{upd}^{\circlearrowleft}$ to **0** for $|G| - 1$ times, is the N-ary vector $(|G| - 1, ..., |G| - 1)$,
- $\mathrm{wdh}_{\mathbf{En}}(e_{\mathcal{G}})$, the widest anti-chain under $e_{\mathcal{G}}$, can be bounded $O(|G|^{N-1})$.

This yields:

$$\begin{aligned} & O(|G|^2 \cdot o_{\rightarrowtail} \cdot (\mathrm{wdh}_{\mathbf{En}}(e_{\mathcal{G}}))^2 \cdot (t_{upd^{\circlearrowleft}} + t_{\mathrm{sup}} + \mathrm{wdh}_{\mathbf{En}}(e_{\mathcal{G}}) \cdot t_{\leq})) \\ = \; & O(|G|^2 \cdot o_{\rightarrowtail} \cdot (|G|^{N-1})^2 \cdot (N^2 + N + |G|^{N-1} \cdot N)) \\ = \; & O(o_{\rightarrowtail} \cdot |G|^{2N+2-2} \cdot (N^2 + |G|^{N-1} \cdot N)) \\ = \; & O(o_{\rightarrowtail} \cdot |G|^{2N} \cdot (N^2 + |G|^{N-1} \cdot N)) \end{aligned}$$

For space complexity, we obtain:

$$\begin{aligned} & O(|G| \cdot \mathrm{wdh}_{\mathbf{En}}(e_{\mathcal{G}}) \cdot N) \\ & O(|G| \cdot |G|^{N-1} \cdot N) \\ & O(|G|^N \cdot N) \end{aligned}$$

☐

Definition 4.13. For N-dimensional energy games, we define the *flattened energies* as $\widehat{\mathbf{En}} := (\{0, 1, \infty\})^N$. A standard energy $e \in \mathbf{En}$ is cast to $\widehat{\mathbf{En}}$ 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* monotonic energy game winner problem the variant of Definition 4.9 that outputs

$$\widehat{\mathsf{Win}_{\mathrm{a}}^{\min}}(g) := \mathsf{Min}(\mathsf{Win}_{\mathrm{a}}(g) \cap \widehat{\mathbf{En}}).$$

Clearly, Algorithm 4.1 can too be used to solve the flattened version of the problem by adapting $\mathsf{upd}^{\circlearrowleft}$ to represent all components above 1 by ∞. Effectively, this decouples the size of Pareto fronts from the game size. In this instance, Lemma 4.6 becomes:

Lemma 4.7. *Algorithm 4.1 solves the* flattened *energy game problem on an N-dimensional declining energy game $\mathcal{G}^{\lightning} = (G, G_{\mathrm{d}}, \rightarrowtail, w)$ in time*

$$O(o_{\rightarrowtail} \cdot |G|^{2N} \cdot (N^2 + 3^{N-1} \cdot N))$$

and in space $O(|G| \cdot 3^{N-1} \cdot N)$.

The bounds we have established are *exponential* with regard to the dimensionality N. 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.

4.3.5 Polynomial Spectroscopy Complexity

We have gathered enough material to have this section prove that the spectroscopy problem for the polynomial-time strong spectrum is indeed solvable in polynomial time, thus justifying the name we chose for the spectrum.

Theorem 4.4. *Given a transition system $\mathcal{S}$, the spectroscopy problem for the $\mathsf{N}^{\mathsf{poly-strong}}$-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 $\mathsf{N}^{\mathsf{poly-strong}}$-spectrum by deciding the winning budgets of the bisimulation energy game $\mathcal{G}^{\lightning}_{\mathrm{B}}$ on $\mathcal{S} = (\mathcal{P}, \mathit{Act}, \rightarrow)$. Thus, all that remains to be done is to instantiate the winning budget complexity of Lemma 4.6 for the case $N = 3$ with the size of $\mathcal{G}^{\lightning}_{\mathrm{B}}$ according to Definition 4.7.

The amount 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 amount of defender positions, which again can be left by $|\mathcal{P}| \cdot |\rightarrow|$ moves.

In total, this amounts to $o_{\rightarrowtail^{\lightning}_{\mathrm{B}}}$ in $O(|\mathcal{P}| \cdot |\rightarrow|)$ and to $|G^{\lightning}_{\mathrm{B}}|$ in $O(|\mathcal{P}|^2)$.

Inserting the parameters in the time bounds of Lemma 4.6 yields:

$$\begin{array}{lllllll} & O(& o_{\rightarrowtail} & \cdot & |G|^{2N} & \cdot & (N^2 + |G|^{N-1} \cdot N) &) \\ = & O(& (|\mathcal{P}| \cdot |\rightarrow|) & \cdot & (|\mathcal{P}|^2)^{2 \cdot 3} & \cdot & (3^2 + (|\mathcal{P}|^2)^{3-1} \cdot 3) &) \\ = & O(& (|\mathcal{P}| \cdot |\rightarrow|) & \cdot & |\mathcal{P}|^{12} & \cdot & |\mathcal{P}|^4 &) \\ = & O(& |\mathcal{P}|^{17} \cdot |\rightarrow| &). \end{array}$$

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 4.7. □

4.4 Discussion

This chapter has shown how to solve the spectroscopy problem for the $\mathsf{N}^{\mathsf{poly-strong}}$ spectrum in polynomial time, thereby deciding all its preorders and equivalences *at once*.

The core ingredient has been to characterize the spectrum's observation languages through an energy game (Note 5, Theorem 4.1) derived from the bisimulation game. The Pareto front view is slightly more general than other characteristic games, which Hüttel & Shukla (1996) for instance use to establish polynomial upper bounds individually for simulation-like notions.

To compute the Pareto fronts, we use a fixed point algorithm (Note 6, Algorithm 4.1). It can be thought of as a generalization of well-known ways to compute shortest distances in graphs:

Remark 4.4. 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 $(\mathbb{Z}, \leq)$ where only deadlock positions belong to the defender and where all updates constitute one-dimensional integer addition. Then, $e \in \mathsf{Win}_{\mathsf{a}}^{\min}(g)$ iff e is the length of a shortest path from g to some defender position (where negative $(-z)$-updates stand for what is usually written as a positive edge weight z for graphs). In this instance, Algorithm 4.1 behaves like the well-known Bellman–Ford algorithm for shortest paths.

Algorithm 4.1 contains relevant generalizations and practical improvements compared to the one originally used in Bisping (2023a). 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.

Knowing 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 PSPACE = 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

Todo

This section will be filled with contents from Process Equivalence Problems as Energy Games.

5.1 The Strong Spectroscopy Game

5.1.1 The Game

Definition 5.1. For a system $\mathcal{S} = (\mathcal{P}, \mathit{Act}, \rightarrow)$, the 6-dimensional *spectroscopy energy game* $\mathcal{G}^{\mathcal{S}}_{\triangle}[g_0, e_0] = (G, G_{\mathrm{d}}, \rightarrowtail, w, g_0, e_0)$ consists of

- *attacker positions* $[p, Q]_{\mathrm{a}} \in G_{\mathrm{a}}$,
- *attacker clause positions* $[p, q]^{\wedge}_{\mathrm{a}} \in G_{\mathrm{a}}$,
- *defender conjunction positions* $(p, Q, Q_*)_{\mathrm{d}} \in G_{\mathrm{d}}$,

where $p, q \in \mathcal{P}$ and $Q, Q_* \in 2^{\mathcal{P}}$, and six kinds of moves:

$$\begin{array}{rrcll}
\textit{observation} & [p, Q]_{\mathrm{a}} & \xrightarrow{-1,0,0,0,0,0} & [p', Q']_{\mathrm{a}} & \text{if } p \xrightarrow{\alpha} p', Q \xrightarrow{\alpha} Q', \\
\textit{conjunction} & [p, Q]_{\mathrm{a}} & \xrightarrow{0,0,0,0,0,0} & (p, Q \setminus Q_*, Q_*)_{\mathrm{d}} & \text{if } Q_* \subseteq Q, \\
\textit{conj. revival} & (p, Q, Q_*)_{\mathrm{d}} & \xrightarrow{\mathtt{min}_{\{1,3\}},-1,0,0,0,0} & [p, Q_*]_{\mathrm{a}} & \text{if } Q_* \neq \emptyset, \\
\textit{conj. answer} & (p, Q, Q_*)_{\mathrm{d}} & \xrightarrow{0,-1,0,\mathtt{min}_{\{3,4\}},0,0} & [p, q]^{\wedge}_{\mathrm{a}} & \text{if } q \in Q, \\
\textit{positive conjunct} & [p, q]^{\wedge}_{\mathrm{a}} & \xrightarrow{\mathtt{min}_{\{1,4\}},0,0,0,0,0} & [p, \{q\}]_{\mathrm{a}}, & \text{and} \\
\textit{negative conjunct} & [p, q]^{\wedge}_{\mathrm{a}} & \xrightarrow{\mathtt{min}_{\{1,5\}},0,0,0,0,-1} & [q, \{p\}]_{\mathrm{a}} & \text{if } p \neq q.
\end{array}$$

A schematic representation can be seen in Figure 5.1.

Lemma 5.1. *p_0 and q_0 are bisimilar iff the defender wins $\mathcal{G}_{\triangle}$ from $[p_0, \{q_0\}]_{\mathrm{a}}$ with e_0 for every initial energy budget e_0, that is, if* $(\infty, \infty, \infty, \infty, \infty, \infty) \in \mathsf{Win}^{\max}_{\mathrm{d}}([p_0, \{q_0\}]_{\mathrm{a}})$.[68]

[68] Proof in Lemma 1 of Bisping (2023b).

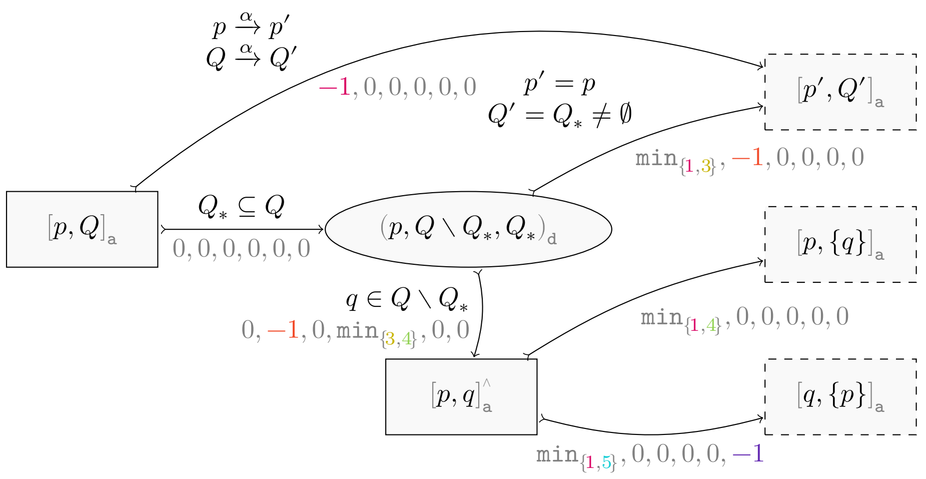

$p \xrightarrow{\alpha} p'$
$Q \xrightarrow{\alpha} Q'$
$-1, 0, 0, 0, 0, 0$
$p' = p$
$Q' = Q_* \neq \emptyset$
$[p', Q']_a$
$\min_{\{1,3\}}, -1, 0, 0, 0, 0$
$[p, Q]_a$
$Q_* \subseteq Q$
$0, 0, 0, 0, 0, 0$
$(p, Q \setminus Q_*, Q_*)_d$
$[p, \{q\}]_a$
$q \in Q \setminus Q_*$
$0, -1, 0, \min_{\{3,4\}}, 0, 0$
$\min_{\{1,4\}}, 0, 0, 0, 0, 0$
$[p, q]^\wedge_a$
$[q, \{p\}]_a$
$\min_{\{1,5\}}, 0, 0, 0, 0, -1$

Figure 5.1: Schematic spectroscopy game $\mathcal{G}_\triangle$ of Definition 5.1.

5.1.2 Correctness

5.1.3 Complexity

5.2 More Specific Games

5.2.1 The Clever Game

5.2.2 Instance Games

5.3 Discussion

- Mention option of non-perfect attacker information as alternative to subset construction
- Difference to Bisping et al. (2022):
  - Lower exponentiality in conjunctions
  - No explicit construction of formulas

6 Spectroscopy for the Weak Silent-Step Spectrum

ℹ Todo

This section will be filled, once the journal version of One Energy Game for the Spectrum between Branching Bisimilarity and Weak Trace Semantics is ready.

6.1 Lifting to the Spectrum of Stable Equivalences

6.1.1 Stable Behavioral Equivalences

6.1.2 The Adapted Spectroscopy Game

6.2 The Weak Spectrum

6.2.1 Special Weak Equivalences

6.2.2 HML with Silent Behavior

6.2.3 Expressing the Weak Spectrum with Quantities

6.3 The Weak Spectroscopy Game

6.3.1 The Game

6.3.2 Correctness

6.4 Discussion

- Mention to add back in strong observations

7 Implementations

7.1 Prototype Implementation: *equiv.io*

7.2 GPU Implementation: *gpuequiv*

Vogel (2024)

7.3 Other Implementations

7.3.1 Computer Game: *The Spectroscopy Invaders*

Trzeciakiewicz (2021)

7.3.2 CAAL Extension

Ozegowski (2023) mention Timo

8 Conclusion

There is a rich body of work on *process equivalence checking* and *refinement checking* for programs and specifications. This thesis argues that most of it can be viewed through one unified lense of *finding differences* in behavior.

References

Adler, R. (2022). *Simulation fehlertoleranter Konsensalgorithmen in Hash.ai* [Bachelor's Thesis]. Technische Universität Berlin.

Alshukairi, Z. (2022). *Automatisierte Reduktion von reaktiver zu starker Bisimilarität* [Bachelor's Thesis]. Technische Universität Berlin.

Babai, L. (2016). Graph isomorphism in quasipolynomial time [extended abstract]. *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, 684–697. https://doi.org/10.1145/2897518.2897542

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

Bell, C. J. (2013). Certifiably sound parallelizing transformations. In *International conference on certified programs and proofs* (pp. 227–242). Springer.

Bisping, B. (2018). *Computing coupled similarity* [Master's thesis, Technische Universität Berlin]. https://coupledsim.bbisping.de/bisping_computingCoupledSimilarity_thesis.pdf

Bisping, B. (2023a). 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. (2023b). *Process equivalence problems as energy games*. 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.), *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.), *Proceedings Combined 31st International Workshop on expressiveness in concurrency and 21st Workshop on structural operational semantics, Calgary, Canada, 9th September 2024* (Vol. 412, pp. 71–88). Open Publishing Association. https://doi.org/10.4204/EPTCS.412.6

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

Bisping, B., & Montanari, L. (2021). A game characterization for contrasimilarity. In O. Dardha & V. Castiglioni (Eds.), *Proceedings Combined 28th International Workshop on expressiveness in concurrency and 18th Workshop *on structural operational semantics* (Vol. 339, 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.), *Tools and algorithms for the construction and analysis of systems: TACAS* (Vol. 11427, 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.), *Tools and algorithms for the construction and analysis of systems: TACAS* (Vol. 12651, 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

Bulik, J. (2021). *Statically analysing inter-process communication in Elixir programs* [Bachelor's Thesis, Technische Universität Berlin]. https://gitlab.com/MrGreenTea/stille-post

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., Koslowski, 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

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

Glabbeek, R. J. van. (1990). The linear time–branching time spectrum: Extended abstract. In J. C. M. Baeten & J. W. Klop (Eds.), *CONCUR'90* (Vol. 458, pp. 278–297). Springer. https://doi.org/10.1007/BFb0039066

Glabbeek, R. J. van. (1993). The linear time–branching time spectrum II: The semantics of sequential systems with silent moves; extended abstract. In E. Best (Ed.), *CONCUR'93* (Vol. 715, pp. 66–81). Springer. https://doi.org/10.1007/3-540-57208-2_6

Glabbeek, R. J. van. (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

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., & 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

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.), *Automata, languages and programming* (Vol. 85, pp. 299–309). Springer. https://doi.org/10.1007/3-540-10003-2_79

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

Kruskal, J. B. (1972). The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory, Series A*, *13*(3), 297–305. 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. Rodriguez-Artalejo (Eds.), *Computer science logic: CSL* (Vol. 1683, 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.), *33rd international conference on concurrency theory (CONCUR 2022)* (Vol. 243, 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.

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.

Lönne, B. (2023). *An educational computer game about counterfactual truth conditions* [Bachelor's Thesis, Technische Universität Berlin].

https://github.com/Brunololos/counterfactuals-demo

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

Milner, R. (1989). *Communication and concurrency*. Prentice-Hall, Inc.

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.

Peacock, D. A. (2020). *Process equivalences as a video game* [Bachelor's Thesis, Technische Universität Berlin]. https://drive.google.com/drive/folders/19YuOCXI_7CEB4p9fgr6kDKCTbFwx2VU9

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

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. (2012). *Introduction to bisimulation and coinduction*. Cambridge University Press. https://doi.org/10.1017/CBO9780511777110

Stirling, C. (1996). Modal and temporal logics for processes. In F. Moller & G. Birtwistle (Eds.), *Logics for concurrency: Structure versus automata* (Vol. 1043, 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.

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

Vogel, G. (2024). *Accelerating process equivalence energy games using We-*

bGPU [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

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). 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/https://doi.org/10.1016/S0304-3975(98)00009-7