Concurrency Theory

9. Lecture: (Un-)Decidability of Bisimilarity*

Dr. Stephan Mennicke

Institute for Theoretical Computer Science
Knowledge-Based Systems Group

June 16, 2025

*The Infinte Case

International Center
for Computational Logic

Recall and Conclude

Theorem 9.1 CCS is Turing-complete. For every Minsky machine $\mathcal{M}$ there is a process $P(\mathcal{M})$ with a special constant L_H representing the halting line of $\mathcal{M}$ such that $\mathcal{M}$ terminates if and only if $P(\mathcal{M}) \xrightarrow{\tau}{}^{\star} \xrightarrow{\checkmark}$.

The Bisimilarity Problem

Input Processes $p, q \in \mathsf{Pr}$.

Output Yes if and only if $p \simeq q$.

Theorem 9.2 $\simeq$ is undecidable for CCS processes.

Recall: What makes $\simeq$ undecidable?

Our simulation of Minsky machines made heavy use of

1. process constants (to mimic recursion)
2. synchronization (to enforce proper moves) (**today**)

Theorem 9.3 (Theorem 36 in Lecture 7) $\simeq$ is P-complete for CCS_{fin} processes.

- membership in P: easy
- hardness: by reduction from MCVP (*defender's forcing* teachnique)

Recall: Process Rewrite Systems (PRS)

Let $\mathcal{K}$ be a set of *process constants*. We define four classes of process expressions:

$$\mathbf{1} : E ::= \varepsilon \mid X$$

$$\mathcal{P} : E ::= \varepsilon \mid X \mid E \parallel E$$

$$\mathcal{S} : E ::= \varepsilon \mid X \mid E.E$$

$$\mathcal{G} : E ::= \varepsilon \mid X \mid E \parallel E \mid E.E$$

where ε is the *empty process* and $X \in \mathcal{K}$.

- The operator . stands for *sequential composition* ($\mathcal{S}$ for sequential process expressions)

Recall: Process Rewrite Systems (PRS)

- The operator $\|$ stands for *parallel composition* ($\mathcal{P}$ for parallel process expressions)

We do not distinguish process expressions related by *structural congruence*, being the smallest congruence such that . is associative, $\|$ is associative and commutative, and ε is a unit for . and $\|$.

Recall: Process Rewrite Systems (PRS)

Definition 9.4 Let $\alpha, \beta \in \{\mathbf{1}, \mathcal{P}, \mathcal{S}, \mathcal{G}\}$ such that $\alpha \subseteq \beta$ and let Act be a set of *actions*. An (α, β)-*PRS* is a finite set $\Delta \subseteq (\alpha \setminus \{\varepsilon\}) \times \mathsf{Act} \times \beta$ of rewrite rules, written $E \xrightarrow{a} F$ for $(E, a, F) \in \Delta$. Such a PRS determines a labeled transition system $\mathcal{T}(\Delta)$ with $\longrightarrow$ being the smallest transition relation satisfying the following rules

$$\frac{\left(E \xrightarrow{a} E'\right) \in \Delta}{E \xrightarrow{a} E'} \qquad \frac{E \xrightarrow{a} E'}{E.F \xrightarrow{a} E'.F} \qquad \frac{E \xrightarrow{a} E'}{E \parallel F \xrightarrow{a} E' \parallel F}$$

Recall: Basic Parallel Processes (BPP)

Equivalently, $(\mathbf{1}, \mathcal{P})$-PRSs. The fragment of CCS without restriction, relabeling and communication (i.e., sychronization). For example,

$$X \xrightarrow{a} X \parallel Y \parallel Y \text{ and } Y \xrightarrow{b} \varepsilon$$

allow for the transitions

$$\xrightarrow{a} X \parallel Y \parallel Y \xrightarrow{b} X \parallel Y \xrightarrow{a} X \parallel Y \parallel Y \parallel Y \xrightarrow{b} X \parallel Y \parallel Y \xrightarrow{a} X \parallel Y \parallel Y \parallel Y$$

Every BPP expression E (over Δ) can be viewed as a

- *Parikh vector* $\varphi(E) = (k_1, k_2, \ldots, k_n) \in \mathbb{N}^n$
- in which we assume the constants used in Delta are assumed to be ordered

Recall: Basic Parallel Processes (BPP)

- e.g., like $X_1, X_2, ..., X_n$;
- then k_j $(1 \leq j \leq n)$ counts the number of occurrences of X_j in E.

Recall: An Example BPP System

$$X_1 \xrightarrow{a} X_1 \parallel X_2 \qquad (1,0,0,0) \xrightarrow{a} (1,1,0,0)$$

$$X_2 \xrightarrow{b} X_3 \qquad (0,1,0,0) \xrightarrow{b} (0,0,1,0)$$

$$X_3 \xrightarrow{b} \varepsilon \qquad (0,0,1,0) \xrightarrow{b} (0,0,0,0)$$

$$X_4 \xrightarrow{a} X_4 \parallel X_3 \parallel X_3 \qquad (0,0,0,1) \xrightarrow{a} (0,0,2,1)$$

The sequence of Transitions

$$X_1 \xrightarrow{a} X_1 \parallel X_2 \xrightarrow{a} X_1 \parallel X_2 \parallel X_2 \xrightarrow{b} X_1 \parallel X_2 \parallel X_3 \xrightarrow{b} X_1 \parallel X_2$$

can analogously be represented by

Recall: An Example BPP System

$$(1,0,0,0) \xrightarrow{a} (1,1,0,0) \xrightarrow{a} (1,2,0,0) \xrightarrow{b} (1,1,1,0) \xrightarrow{b} (1,1,0,0)$$

The Bisimilarity Problem of PRSs

Given two processes E and F from some class of the PRS hierarchy, are they bisimilar?

Theorem 9.5 It is decidable whether $E \simeq F$ for any two BPP processes E and F.

The Tableau Algorithm for BPP

Let E and F be two BPP processes over Δ.

A **tableau** for E and F is a maximal *proof tree* rooted in $(\varphi(E), \varphi(F))$ and built according to the following rules.

The Tableau Algorithm for BPP

Let E and F be two BPP processes over Δ.

A **tableau** for E and F is a maximal *proof tree* rooted in $(\varphi(E), \varphi(F))$ and built according to the following rules.

Let $(\alpha, \beta) \in \mathbb{N}^{2n}$ be a node in the tree.

(α, β) is either *terminal (leaf)* or *nonterminal*. The following nodes are terminal:

- (α, α) is a *successful leaf* for any $\alpha \in \mathbb{N}^m$,
- (α, β) is a *successful leaf* if α and β are **deadlocks**,
- (α, β) is an *unsuccessful leaf* if, for some $a \in \mathsf{Act}$, $\alpha \xrightarrow{a}$ and $\beta \not\xrightarrow{a}$ or $\beta \xrightarrow{a}$ and $\alpha \not\xrightarrow{a}$.

The Tableau Algorithm for BPP

A node is an *ancestor* of (α, β) if it is on the path from root to (α, β) and at least one application of the rule EXPAND (to be defined) separates them.

If (α, β) is not a leaf, then we *reduce it* using the following rules:

$$\text{RED}_{\text{L}} \frac{(\alpha, \beta)}{(\gamma + \omega, \beta)}$$

if there is an ancestor (γ, δ) or (δ, γ) of (α, β) such that $\gamma <_\ell \delta$ and $\alpha = \delta + \omega$ for some $\omega \in \mathbb{N}^n$

$$\text{RED}_{\text{R}} \frac{(\alpha, \beta)}{(\alpha, \gamma + \omega)}$$

if there is an ancestor (γ, δ) or (δ, γ) of (α, β) such that $\gamma <_\ell \delta$ and $\beta = \delta + \omega$ for some $\omega \in \mathbb{N}^n$

The Tableau Algorithm for BPP

If no reduction is applicable to (α, β) and it is not a leaf node, apply EXPAND for a set of relations S_a, $a \in \mathsf{Act}$, where

$$S_a \subseteq \left\{ \alpha' \,\middle|\, \alpha \xrightarrow{a} \alpha' \right\} \times \left\{ \beta' \,\middle|\, \beta \xrightarrow{a} \beta' \right\}$$

such that

- for each α' s.t. $\alpha \xrightarrow{a} \alpha'$, there is some β' such that $(\alpha', \beta') \in S_a$ and
- for each β' s.t. $\beta \xrightarrow{a} \beta'$, there is some α' such that $(\alpha', \beta') \in S_a$.

$$\text{EXPAND}\,\frac{(\alpha, \beta)}{\{(\alpha', \beta') \mid a \in \mathsf{Act} \wedge (\alpha', \beta') \in S_a\}}$$

The Tableau Algorithm for BPP

We call a tableau *successful* if it is maximal (i.e., no more rules are applicable) and all its leaves are successful.

Lemma 9.6 Any tableau for BPP processes E and F is finite and there are only finitely many tableaux for E and F.

Lemma 9.7 Let E and F be BPP processes over Δ. If $E \simeq F$, then there is a successful tableau for E and F.

The Tableau Algorithm for BPP

Lemma 9.8 Let E and F be BPP processes over Δ. If there is a successful tableau for E and F, then $E \simeq F$.

Recall: Example BPP System

$$X_1 \xrightarrow{a} X_1 \parallel X_2 \qquad (1,0,0,0) \xrightarrow{a} (1,1,0,0)$$

$$X_2 \xrightarrow{b} X_3 \qquad (0,1,0,0) \xrightarrow{b} (0,0,1,0)$$

$$X_3 \xrightarrow{b} \varepsilon \qquad (0,0,1,0) \xrightarrow{b} (0,0,0,0)$$

$$X_4 \xrightarrow{a} X_4 \parallel X_3 \parallel X_3 \qquad (0,0,0,1) \xrightarrow{a} (0,0,2,1)$$

Want to show that $X_1 \simeq X_4$, meaning $1000 \simeq 0001$.

Starting from the root node, $(1000, 0001) = (\delta, \gamma)$, observe that no reduction can take place (**hint:** no ancestors yet): $(1100, 0001) = (\alpha, \beta)$

$$\gamma <_\ell \delta \text{ and } \alpha = \delta + 0100$$

Recall: Example BPP System

$$\dfrac{\dfrac{\dfrac{\dfrac{(1000, 0001)}{\mathrm{EXPAND}}}{(1100, 0021)}}{\mathrm{RED_L}}}{\dfrac{\dfrac{(0101, 0021)}{\mathrm{EXPAND}}}{\dfrac{(0121, 0041)}{\mathrm{RED_L}}}} \quad (0011, 0011)}{(0041, 0041)}$$

Tableau Algorithm: Completeness

Lemma 9.7 Let E and F be BPP processes over Δ. If $E \simeq F$, then there is a successful tableau for E and F.

We construct a successful tableau inductively for $(\varphi(E), \varphi(F))$ such that every node (α, β) in the tableau satisfies $\alpha \simeq \beta$ (hence, no unsuccessful leaves).

Tableau Algorithm: Completeness

Let (α, β) be a node such that $\alpha \simeq \beta$ and consider an application of $\mathrm{RED}_{\mathrm{L}}$ for ancestor (γ, δ):

$$\mathrm{RED}_{\mathrm{L}} \frac{(\alpha, \beta)}{(\gamma + \omega, \beta)}$$

if there is an ancestor (γ, δ) or (δ, γ) of (α, β) such that $\gamma <_\ell \delta$ and $\alpha = \delta + \omega$ for some $\omega \in \mathbb{N}^n$

By induction hypothesis, $\gamma \simeq \delta$. Consequently by $\simeq$ being a congruence,

$$(\gamma + \omega) \simeq (\delta + \omega) = \alpha \simeq \beta$$

Similarly for $\mathrm{RED}_{\mathrm{R}}$.

Tableau Algorithm: Completeness

For EXPAND for some

$$S_a \subseteq \left\{ \alpha' \,\middle|\, \alpha \xrightarrow{a} \alpha' \right\} \times \left\{ \beta' \,\middle|\, \beta \xrightarrow{a} \beta' \right\}$$

such that

- for each α' s.t. $\alpha \xrightarrow{a} \alpha'$, there is some β' such that $(\alpha', \beta') \in S_a$ and
- for each β' s.t. $\beta \xrightarrow{a} \beta'$, there is some α' such that $(\alpha', \beta') \in S_a$.

$$\text{EXPAND} \frac{(\alpha, \beta)}{\{(\alpha', \beta') \mid a \in \mathsf{Act} \wedge (\alpha', \beta') \in S_a\}}$$

Thus, we can choose for every $a \in \mathsf{Act}$ a relation S_a such that $(\alpha', \beta') \in S_a$ implies $\alpha' \simeq \beta'$.

Tableau Algorithm: Completeness

Open Questions

1. When does the construction stop?
2. Are we guaranteed to find a successful tableau?

Two Questions – One Answer

Lemma 9.6 Any tableau for BPP processes E and F is finite and there are only finitely many tableaux for E and F.

Two Questions – One Answer

Lemma 9.6 Any tableau for BPP processes E and F is finite and there are only finitely many tableaux for E and F.

Thus, every candidate construction as in Lemma 9.7 must terminate and there are only finitely many different ways.

1. Every tableau for E and F is finitely branching because
   - E and F may use finitely many actions (Δ is finite),
   - any relation S_a is finite, and
   - there are finitely many such relations.
2. Suppose, the tableau is infinite,
   - there must be an infinite branch from $\mathbb{N}^{2n}$

Two Questions – One Answer

- RED rules can be applied only finitely often in a row
- thus, there is an infinite subsequence with EXPAND rule applications
- by Dickson's Lemma, there must be a nondecreasing, yet infinite, subsequence

$$(\alpha_1, \beta_1) \leq_c (\alpha_2, \beta_2) \leq_c \ldots$$

- EXPAND cannot be applied to (α_2, β_2) since RED is applicable. **Contradiction**

3. As there are finitely many S_a for EXPAND and finitely many possiblities of the RED rule, there are finitely many ways to extend an initial partial tableau;

Two Questions – One Answer

4. If there were infinitely many tableaux for $(\varphi(E), \varphi(F))$, there must be one tableau with an infinite branch. **Contradiction**

Tableau Algorithm: Soundness

Lemma 9.8 Let E and F be BPP processes over Δ. If there is a successful tableau for E and F, then $E \simeq F$.

Suppose, there is a successful tableau for E and F, but $E \not\simeq F$. Then we can construct path from the root $(\varphi(E), \varphi(F))$ to some leaf, such that for all pairs (α, β) on that path, $\alpha \not\simeq \beta$ holds.

If $E \not\simeq F$, then there is a $k \in \mathbb{N}$ such that $E \not\simeq_k F$ such that k is minimal.

Note, if $\alpha \not\simeq_k \beta$ such that k is minimal and we apply EXPAND, then at least one child (α', β') satisfies $\alpha' \not\simeq_{k-1} \beta'$. We pick such a child to expand our path from the root.

Tableau Algorithm: Soundness

$$\text{RED}_\text{L}\frac{(\alpha, \beta)}{(\gamma + \omega, \beta)}$$ if there is an ancestor (γ, δ) or (δ, γ) of (α, β) such that $\gamma <_\ell \delta$ and $\alpha = \delta + \omega$ for some $\omega \in \mathbb{N}^n$

If we apply RED_L on (α, β) where $\alpha \not\simeq_k \beta$ and k is minimal, then

- ancestor (γ, δ) is separated by at least one application of EXPAND
- so, $\gamma \simeq_k \delta$, and
- $(\gamma + \omega) \not\simeq_k \beta$ (hint: $\simeq_k$ is a congruence for BPP)

Same argument works for RED_R. Hence, there must be a path from the root to some leaf such that every node (α, β) on that path satisfies $\alpha \not\simeq \beta$.

Contradiction to the fact that the path contains a successful leaf.

Tableau Algorithm: Summary

Theorem 9.5 It is decidable whether $E \simeq F$ for any two BPP processes E and F.

Proof. By the tableau algorithm together with Lemma 9.7 and Lemma 9.8. ■

- Complexity of the tableau algorithm unknown; so far, no primitive recursive upper bound
- Using a different algorithm, PSpace-completeness of bisimilarity for BPPs established
- Similar algorithms have been found for further PRS classes like BPA

Petri Nets (PNs)

Equivalently, $(\mathcal{P}, \mathcal{P})$-PRSs. PNs add the feature of synchronization to BPP.

Theorem 9.9 It is undecidable if two Petri nets E and F are bisimilar.

This theorem shows undecidability of bisimilarity for CCS processes (cf. Theorem 9.2).

Recall: Minsky Machines

Definition 9.10 A *Minsky machine* is a pair $\mathcal{M} = (\mathbb{P}, R)$ with $\mathbb{P}$ a finite sequence $\ell_1\ell_2...\ell_m$ of *program lines* and a finite set R of *counters* (i.e., $R = \{c_1, c_2, ..., c_n\}$) such that ℓ_m : HALT and each line ℓ_i $(1 \leq i < m)$ has one of the following shapes

$$i : \mathsf{inc}\ c : j \qquad\qquad i : \mathsf{dec}\ c : j : k$$

for counter $c \in R$ and $1 \leq j, k \leq m$.

A configuration of $\mathcal{M}$ is a pair $\gamma = (i, \beta)$ where $1 \leq i \leq m$ and $\beta : R \to \mathbb{N}$. For $\gamma_1 = (i, \beta_1)$ and $\gamma_2 = (j, \beta_2)$, define $\gamma_1 \triangleright \gamma_2$ by cases:

$\boldsymbol{i : \mathsf{inc}\ c : j}$ $\beta_2 = \beta_1[c \mapsto \beta_1(c) + 1]$

$\boldsymbol{i : \mathsf{dec}\ c : k_1 : k_2}$ if $\beta_1(c) > 0$, $j = k_1$ and $\beta_2 = \beta_1[\beta_1(c) - 1]$; otherwise, if $\beta_1(c) = 0$, $\beta_2 = \beta_1$ and $j = k_2$.

Recall: Minsky Machines

Theorem 9.11 The *halting problem* for 2-counter Minsky machines is undecidable.

Implementing Minsky Machines with Petri Nets

Our BPP for a Minsky machine will contain process constants C, one for every counter of the program. Also, every line ℓ_i of the program will be represented by a process constant L_i.

m : HALT $L_m \xrightarrow{h} \varepsilon$

i : inc c : j we get $L_i \xrightarrow{i} C \mid L_j$

i : dec c : k_1 : k_2 we get $L_i \mid C \xrightarrow{d} L_{k_1}$ and $L_i \xrightarrow{z} L_{k_2}$.

Now we get that a Minsky machine halts if and only if its BPP representation

1. exhibits some trace ending in h
2. has finite state space

Where is the problem?

Decidable Problems for Petri Nets

It is *decidable* if

1. a Petri net ever reaches a given state;
2. a Petri net *covers* a given process constant; e.g., L_m
3. a Petri net ever exhibits a trace including a certain symbol; e.g., h
4. a Petri net has finite behavior (i.e., finitely reachable states);
5. ...

It is *undecidable* if

1. two Petri nets N_1 and N_2 are bisimilar.

Reviewing our Construction

Our BPP for a Minsky machine will contain process constants C_n, one for every counter of the program. Also, every line ℓ_i of the program will be represented by a process constant L_i.

m : **HALT** $L_m \xrightarrow{h} \varepsilon$

i : **inc** c : j we get $L_i \xrightarrow{i} C \mid L_j$

i : **dec** c : k_1 : k_2 we get

- $L_i \mid C \xrightarrow{d} L_{k_1}$,
- $L_i \xrightarrow{z} L_{k_2}$, and
- $L_i \mid C \xrightarrow{z} L_{k_2} \mid C$ (**cheating**)

How to detect **cheating**? By bisimilarity 😀

An Exercise in Defender's Forcing

For counters c_1, c_2 and the m lines of the program, we use the process constants $\mathcal{K} = \{C_1, C_2, L_1, L_2, ..., L_m, L'_1, L'_2, ..., L'_m\}$ and define Δ:

$$L_m \xrightarrow{h} \varepsilon \qquad\qquad m : \mathsf{HALT}$$

$$L_i \xrightarrow{i} L_j \mid C \qquad L'_i \xrightarrow{i} L'_j \mid C \qquad\qquad i : \mathsf{inc}\ c : j$$

$$L_i \mid C \xrightarrow{d} L_{k_1} \qquad L'_i \mid C \xrightarrow{d} L'_{k_1} \qquad\qquad i : \mathsf{dec}\ c : k_1 : k_2$$

$$L_i \xrightarrow{z} L_{k_2} \qquad L'_i \xrightarrow{z} L'_{k_2}$$

$$L_i \mid C \xrightarrow{z} L'_{k_2} \mid C \qquad L'_i \mid C \xrightarrow{z} L_{k_2} \mid C$$

Bisimilarity is Undecidable for Petri Nets

By this construction, we get that the Minsky machine (with at least two counters) halts if and only if $L_1 \not\approx L_1'$.

For initial counter values $n_1, n_2 \in \mathbb{N}$, $L_1 \mid C_1^{n_1} \mid C_2^{n_2} \not\approx L_1' \mid C_1^{n_1} \mid C_2^{n_2}$.

Theorem 9.9 It is undecidable if two Petri nets E and F are bisimilar.

As a corollary, we get the ultimate answer:

Theorem 9.2 $\simeq$ is undecidable for CCS processes.

Proof of Theorem 9.9

A Minsky machine $\mathcal{M}$ halts (i.e., reaches its last program line) if and only if $L_1 \not\approx L'_1$.

$$L_m \xrightarrow{h} \varepsilon \qquad\qquad m : \mathsf{HALT}$$

$$L_i \xrightarrow{i} L_j \mid C \qquad L'_i \xrightarrow{i} L'_j \mid C \qquad i : \mathsf{inc}\ c : j$$

$$L_i \mid C \xrightarrow{d} L_{k_1} \qquad L'_i \mid C \xrightarrow{d} L'_{k_1} \qquad i : \mathsf{dec}\ c : k_1 : k_2$$

$$L_i \xrightarrow{z} L_{k_2} \qquad L'_i \xrightarrow{z} L'_{k_2}$$

$$L_i \mid C \xrightarrow{z} L'_{k_2} \mid C \qquad L'_i \mid C \xrightarrow{z} L_{k_2} \mid C$$

If $\mathcal{M}$ (eventually) halts, there is a *truthful* simulation of $\mathcal{M}$ $L_1 \longrightarrow^\star L_m$ that con only be answered by $L'_1 \longrightarrow^\star L'_m$. But $L_m \xrightarrow{h}$ and $L'_m \not\xrightarrow{h}$.

What Makes Bisimilarity Undecidable?

Our simulation of Minsky machines made heavy use of

1. process constants (to mimic recursion)
2. synchronization (to enforce proper moves)

Theorem 9.3 (Theorem 36 in Lecture 7) $\simeq$ is P-complete for CCS_{fin} processes.

Theorem 9.5 It is decidable whether $E \simeq F$ for any two BPP processes E and F.

What Makes Bisimilarity Undecidable?

Theorem 9.9 It is undecidable if two Petri nets E and F are bisimilar.

Theorem 9.2 $\simeq$ is undecidable for CCS processes.

Our simulation exhibited non-determinism:

Theorem 9.12 For two deterministic Petri nets E and F, it is decidable if $E \simeq F$.

Decision Procedure for Deterministic Systems

Recall that $\simeq = \equiv_{\mathsf{tr}}$ for deterministic systems.

Thus, deciding bisimilarity amounts to deciding trace equivalence (i.e., trace inclusion).

The proof requires more knowledge about problems for Petri nets that are decidable. In particular, we will deal with

1. the *Boundedness Problem*
2. the *Coverability Problem*
3. the *Liveness Problem*

For Petri nets E and F, we will construct a Petri net $E \oplus F$ such that $E \equiv_{\mathsf{tr}} F$ if and only if $E \oplus F$ is *live*.

Outlook

Have we talked about Petri nets?

Places $\mathcal{K}$

Transitions elements of Δ as *objects*

Arcs Δ

Markings states or processes

Yes, we have talked about Petri nets and directly found their limits.

Next Exercises and Lectures

- graphical notation
- (further) decidability results for Petri nets
- complexity of Petri net counting