Concurrency Theory

Winter 2025/26

Lecture 11: Mutually Recursive Equational Systems

Thomas Noll, Peter Thiemann
Programming Languages Group
University of Freiburg

https://proglang.github.io/teaching/25ws/ct.html

Thomas Noll, Peter Thiemann

Winter 2025/26

Outline of Lecture 11

1. Recap: Hennessy-Milner Logic with Recursion
2. Fixed Points and System Properties
3. Mutually Recursive Equational Systems
4. Characteristic Formulae
5. Mixing Least and Greatest Fixed Points

Syntax of HML with One Recursive Variable

Initially: only one variable (for simplicity; later: mutual recursion)

Definition (Syntax of HML with one variable)

The set HMF_X of Hennessy-Milner formulae with one variable X over a set of actions Act is defined by the following syntax:

$$
\begin{array}{rll}
F ::= & X & \text{(variable)} \\
\mid & \mathrm{tt} & \text{(true)} \\
\mid & \mathrm{ff} & \text{(false)} \\
\mid & F_1 \wedge F_2 & \text{(conjunction)} \\
\mid & F_1 \vee F_2 & \text{(disjunction)} \\
\mid & \langle \alpha \rangle F & \text{(diamond)} \\
\mid & [\alpha] F & \text{(box)}
\end{array}
$$

where $\alpha \in Act$.

Semantics of HML with One Recursive Variable

So far: $[\![F]\!] \subseteq S$ for $F \in \mathit{HMF}$ and LTS $(S, \mathit{Act}, \longrightarrow)$.

Now: Semantics of formula depends on states that (are assumed to) satisfy X ("predicate transformer").

Definition (Semantics of HML with one variable)

Let $(S, \mathit{Act}, \longrightarrow)$ be an LTS and $F \in \mathit{HMF}_X$. The semantics of F,

$$[\![F]\!] : 2^S \to 2^S,$$

is defined by

$$
\begin{aligned}
[\![X]\!](T) &:= T \\
[\![\mathrm{tt}]\!](T) &:= S \\
[\![\mathrm{ff}]\!](T) &:= \emptyset \\
[\![F_1 \wedge F_2]\!](T) &:= [\![F_1]\!](T) \cap [\![F_2]\!](T) \\
[\![F_1 \vee F_2]\!](T) &:= [\![F_1]\!](T) \cup [\![F_2]\!](T) \\
[\![\langle\alpha\rangle F]\!](T) &:= \langle\cdot\alpha\cdot\rangle([\![F]\!](T)) \\
[\![[\alpha]F]\!](T) &:= [\cdot\alpha\cdot]([\![F]\!](T))
\end{aligned}
$$

Lemma

Let $(S, Act, \longrightarrow)$ *be an LTS and* $F \in HMF_X$*. Then*

(1) $[\![F]\!] : 2^S \to 2^S$ *is monotonic w.r.t.* $(2^S, \subseteq)$

(2) $\text{lfp}([\![F]\!]) = \bigcap\{T \subseteq S \mid [\![F]\!](T) \subseteq T\}$

(3) $\text{gfp}([\![F]\!]) = \bigcup\{T \subseteq S \mid T \subseteq [\![F]\!](T)\}$

If, in addition, S *is finite, then*

(4) $\text{lfp}([\![F]\!]) = [\![F]\!]^m(\emptyset)$ *for some* $m \in \mathbb{N}$

(5) $\text{gfp}([\![F]\!]) = [\![F]\!]^M(S)$ *for some* $M \in \mathbb{N}$

Outline of Lecture 11

Greatest Fixed Points and Invariants I

- Invariants (cf. Example 10.5):
  - $Inv(F) \stackrel{max}{=} F \wedge [Act]Inv(F)$ for $F \in HMF$
  - Claim: $s \models Inv(F)$ if all states reachable from s satisfy F

Greatest Fixed Points and Invariants I

- Invariants (cf. Example 10.5):
  - $Inv(F) \stackrel{max}{=} F \wedge [Act]Inv(F)$ for $F \in HMF$
  - Claim: $s \models Inv(F)$ if all states reachable from s satisfy F
- Now: formalise argument and prove its correctness (for arbitrary LTSs)

Greatest Fixed Points and Invariants I

- Invariants (cf. Example 10.5):
  - $Inv(F) \stackrel{max}{=} F \wedge [Act]Inv(F)$ for $F \in HMF$
  - Claim: $s \models Inv(F)$ if all states reachable from s satisfy F
- Now: formalise argument and prove its correctness (for arbitrary LTSs)
- Let $inv : 2^S \to 2^S : T \mapsto [\![F]\!] \cap [\cdot Act \cdot](T)$ be the corresponding semantic function
- By Lemma 10.11(3), $\text{gfp}(inv) = \bigcup\{T \subseteq S \mid T \subseteq inv(T)\}$

Greatest Fixed Points and Invariants I

- Invariants (cf. Example 10.5):
  - $Inv(F) \stackrel{max}{=} F \wedge [Act]Inv(F)$ for $F \in HMF$
  - Claim: $s \models Inv(F)$ if all states reachable from s satisfy F
- Now: formalise argument and prove its correctness (for arbitrary LTSs)
- Let $inv : 2^S \to 2^S : T \mapsto [\![F]\!] \cap [\cdot Act \cdot](T)$ be the corresponding semantic function
- By Lemma 10.11(3), $\mathrm{gfp}(inv) = \bigcup\{T \subseteq S \mid T \subseteq inv(T)\}$
- Direct formulation of invariance property:

$$Inv = \{s \in S \mid \forall w \in Act^*, s' \in S : s \xrightarrow{w} s' \Rightarrow s' \in [\![F]\!]\}$$

Greatest Fixed Points and Invariants I

- Invariants (cf. Example 10.5):
  - $Inv(F) \stackrel{max}{=} F \wedge [Act]Inv(F)$ for $F \in HMF$
  - Claim: $s \models Inv(F)$ if all states reachable from s satisfy F
- Now: formalise argument and prove its correctness (for arbitrary LTSs)
- Let $inv : 2^S \to 2^S : T \mapsto [\![F]\!] \cap [\cdot Act \cdot](T)$ be the corresponding semantic function
- By Lemma 10.11(3), $\text{gfp}(inv) = \bigcup\{T \subseteq S \mid T \subseteq inv(T)\}$
- Direct formulation of invariance property:

$$Inv = \{s \in S \mid \forall w \in Act^*, s' \in S : s \xrightarrow{w} s' \Rightarrow s' \in [\![F]\!]\}$$

Theorem 11.1

For every LTS $(S, Act, \longrightarrow)$, $Inv = \text{gfp}(inv)$ holds.

Greatest Fixed Points and Invariants II

Proof (Theorem 11.1).

Reminder:

- $Inv \overset{(*)}{=} \{s \in S \mid \forall w \in Act^*, s' \in S : s \xrightarrow{w} s' \Rightarrow s' \in [\![F]\!]\}$
- $inv : 2^S \to 2^S : T \mapsto [\![F]\!] \cap [\cdot Act\cdot](T)$
- Lemma 10.11(3): $\text{gfp}(inv) = \bigcup\{T \subseteq S \mid T \subseteq inv(T)\}$

Greatest Fixed Points and Invariants II

Proof (Theorem 11.1).

Reminder:

- $Inv \stackrel{(*)}{=} \{s \in S \mid \forall w \in Act^*, s' \in S : s \xrightarrow{w} s' \Rightarrow s' \in [\![F]\!]\}$
- $inv : 2^S \to 2^S : T \mapsto [\![F]\!] \cap [\cdot Act \cdot](T)$
- Lemma 10.11(3): $\text{gfp}(inv) = \bigcup\{T \subseteq S \mid T \subseteq inv(T)\}$

"$Inv \subseteq \text{gfp}(inv)$":

According to Lemma 10.11(3), it suffices to show that $Inv \subseteq inv(Inv)$.
So let $s \in Inv$. Thus by $(*)$, for all $w \in Act^*$ and $s' \in S$ such that $s \xrightarrow{w} s'$, $s' \in [\![F]\!]$.
We have to show that $s \in inv(Inv)$, which – by definition of inv – is equivalent to (1) $s \in [\![F]\!]$ and (2) $s \in [\cdot Act \cdot](Inv)$:

(1) Choose $w := \varepsilon$ in $(*)$.

(2) To show: for all $\alpha \in Act$, $s' \in S$: if $s \xrightarrow{\alpha} s'$, then $s' \in Inv$.
By $(*)$, $s' \in Inv$ means that for all $\alpha \in Act$, $s', s'' \in S$ and $w' \in Act^*$: if $s' \xrightarrow{w'} s''$, then $s'' \in [\![F]\!]$. Together with $s \xrightarrow{\alpha} s'$, this follows from $(*)$ for $w := \alpha w'$.

Proof (Theorem 11.1).

Reminder:

- $Inv \overset{(*)}{=} \{s \in S \mid \forall w \in Act^*, s' \in S : s \xrightarrow{w} s' \Rightarrow s' \in [\![F]\!]\}$
- $inv : 2^S \to 2^S : T \mapsto [\![F]\!] \cap [\cdot Act\cdot](T)$
- Lemma 10.11(3): $\mathrm{gfp}(inv) = \bigcup\{T \subseteq S \mid T \subseteq inv(T)\}$

"$\mathrm{gfp}(inv) \subseteq Inv$":

Observation: $\mathrm{gfp}(inv) = inv(\mathrm{gfp}(inv)) \overset{(**)}{=} [\![F]\!] \cap [\cdot Act\cdot](\mathrm{gfp}(inv))$.

Let $s \in \mathrm{gfp}(inv)$, $w \in Act^*$ and $s' \in S$ such that $s \xrightarrow{w} s'$.

We show $s' \in [\![F]\!]$ by induction on $|w|$:

(1) $w = \varepsilon$: Here $s = s'$, which implies $s' \in \mathrm{gfp}(inv)$ and thus (by $(**)$) $s' \in [\![F]\!]$.

(2) $w = \alpha w'$: Here $s \xrightarrow{\alpha} s'' \xrightarrow{w'} s'$ for some $s'' \in S$.
Thus, $s'' \in \mathrm{gfp}(inv)$ since $s \in \mathrm{gfp}(inv)$ and by $(**)$.
Therefore, $s' \in [\![F]\!]$ by induction hypothesis for w'.

Altogether, also $s \in Inv$. $\square$

Least Fixed Points and Possibilities

- Possibilities (cf. Example 10.5):
  - $Pos(F) \stackrel{min}{=} F \vee \langle Act \rangle Pos(F)$
  - Claim: $s \models Pos(F)$ if a state satisfying F is reachable from s

Least Fixed Points and Possibilities

- Possibilities (cf. Example 10.5):
  - $Pos(F) \stackrel{min}{=} F \vee \langle Act \rangle Pos(F)$
  - Claim: $s \models Pos(F)$ if a state satisfying F is reachable from s
- Now: formalise argument and prove its correctness (for arbitrary LTSs)

Least Fixed Points and Possibilities

- Possibilities (cf. Example 10.5):
  - $Pos(F) \stackrel{min}{=} F \lor \langle Act \rangle Pos(F)$
  - Claim: $s \models Pos(F)$ if a state satisfying F is reachable from s
- Now: formalise argument and prove its correctness (for arbitrary LTSs)
- Let $pos : 2^S \to 2^S : T \mapsto [\![F]\!] \cup \langle \cdot Act \cdot \rangle (T)$ be the corresponding semantic function
- By Lemma 10.11(2), $\mathrm{lfp}(pos) = \bigcap \{T \subseteq S \mid pos(T) \subseteq T\}$

Least Fixed Points and Possibilities

- Possibilities (cf. Example 10.5):
  - $Pos(F) \stackrel{min}{=} F \vee \langle Act\rangle Pos(F)$
  - Claim: $s \models Pos(F)$ if a state satisfying F is reachable from s
- Now: formalise argument and prove its correctness (for arbitrary LTSs)
- Let $pos : 2^S \to 2^S : T \mapsto [\![F]\!] \cup \langle \cdot Act \cdot \rangle(T)$ be the corresponding semantic function
- By Lemma 10.11(2), $\text{lfp}(pos) = \bigcap\{T \subseteq S \mid pos(T) \subseteq T\}$
- Direct formulation of possibility property:

$$Pos = \{s \in S \mid \exists w \in Act^*, s' \in [\![F]\!] : s \xrightarrow{w} s'\}$$

Least Fixed Points and Possibilities

- Possibilities (cf. Example 10.5):
  - $Pos(F) \stackrel{min}{=} F \vee \langle Act \rangle Pos(F)$
  - Claim: $s \models Pos(F)$ if a state satisfying F is reachable from s
- Now: formalise argument and prove its correctness (for arbitrary LTSs)
- Let $pos : 2^S \to 2^S : T \mapsto [\![F]\!] \cup \langle \cdot Act \cdot \rangle (T)$ be the corresponding semantic function
- By Lemma 10.11(2), $\mathrm{lfp}(pos) = \bigcap \{T \subseteq S \mid pos(T) \subseteq T\}$
- Direct formulation of possibility property:

$$Pos = \{s \in S \mid \exists w \in Act^*, s' \in [\![F]\!] : s \xrightarrow{w} s'\}$$

Theorem 11.2

For every LTS $(S, Act, \longrightarrow)$, $Pos = \mathrm{lfp}(pos)$ holds.

Proof.

similar to Theorem 11.1 □

Outline of Lecture 11

Introducing Several Variables

Sometimes necessary: using more than one variable

Example 11.3

"It is always the case that a process can perform an a-labelled transition leading to a state where b-transitions can be executed forever."

Introducing Several Variables

Sometimes necessary: using more than one variable

Example 11.3

> *"It is always the case that a process can perform an a-labelled transition leading to a state where b-transitions can be executed forever."*

can be specified by

$$Inv(\langle a\rangle Forever(b))$$

where

$$Inv(F) \stackrel{max}{=} F \wedge [Act]Inv(F) \qquad \text{(cf. Theorem 11.1)}$$
$$Forever(b) \stackrel{max}{=} \langle b\rangle Forever(b)$$

Syntax of Mutually Recursive Equational Systems

Definition 11.4 (Syntax of mutually recursive equational systems)

Let $\mathcal{X} = \{X_1, \ldots, X_n\}$ be a set of variables. The set $HMF_{\mathcal{X}}$ of Hennessy-Milner formulae over $\mathcal{X}$ is defined by the following syntax:

$$
\begin{array}{rcll}
F & ::= & X_i & \text{(variable)} \\
& \mid & \mathrm{tt} & \text{(true)} \\
& \mid & \mathrm{ff} & \text{(false)} \\
& \mid & F_1 \wedge F_2 & \text{(conjunction)} \\
& \mid & F_1 \vee F_2 & \text{(disjunction)} \\
& \mid & \langle \alpha \rangle F & \text{(diamond)} \\
& \mid & [\alpha] F & \text{(box)}
\end{array}
$$

where $i \in [n]$ and $\alpha \in Act$. A mutually recursive equational system has the form

$$(X_i = F_{X_i} \mid 1 \leq i \leq n)$$

where $F_{X_i} \in HMF_{\mathcal{X}}$ for every $i \in [n]$.

Semantics of Recursive Equational Systems I

As before: Semantics of formula depends on states satisfying the variables.

Definition 11.5 (Semantics of mutually recursive equational systems)

Let $(S, Act, \longrightarrow)$ be an LTS and $E = (X_i = F_{X_i} \mid 1 \le i \le n)$ a mutually recursive equational system. The semantics of E, $[\![E]\!] : (2^S)^n \to (2^S)^n$, is defined by

$$[\![E]\!](T_1, \ldots, T_n) := ([\![F_{X_1}]\!](T_1, \ldots, T_n), \ldots, [\![F_{X_n}]\!](T_1, \ldots, T_n))$$

where

$$\begin{aligned}
[\![X_i]\!](T_1, \ldots, T_n) &:= T_i \\
[\![\mathrm{tt}]\!](T_1, \ldots, T_n) &:= S \\
[\![\mathrm{ff}]\!](T_1, \ldots, T_n) &:= \emptyset \\
[\![F_1 \wedge F_2]\!](T_1, \ldots, T_n) &:= [\![F_1]\!](T_1, \ldots, T_n) \cap [\![F_2]\!](T_1, \ldots, T_n) \\
[\![F_1 \vee F_2]\!](T_1, \ldots, T_n) &:= [\![F_1]\!](T_1, \ldots, T_n) \cup [\![F_2]\!](T_1, \ldots, T_n) \\
[\![\langle\alpha\rangle F]\!](T_1, \ldots, T_n) &:= \langle\cdot\alpha\cdot\rangle([\![F]\!](T_1, \ldots, T_n)) \\
[\![[\alpha]F]\!](T_1, \ldots, T_n) &:= [\cdot\alpha\cdot]([\![F]\!](T_1, \ldots, T_n))
\end{aligned}$$

Lemma 11.6

Let $(S, Act, \longrightarrow)$ *be a* *finite* *LTS and* $E = (X_i = F_{X_i} \mid 1 \le i \le n)$ *a mutually recursive equational system. Let* $(D, \sqsubseteq)$ *be given by* $D := (2^S)^n$ *and*

$$(T_1, \ldots, T_n) \sqsubseteq (T'_1, \ldots, T'_n) \qquad \textit{iff} \qquad T_i \subseteq T'_i \textit{ for every } i \in [n].$$

Lemma 11.6

Let $(S, Act, \longrightarrow)$ *be a finite LTS and* $E = (X_i = F_{X_i} \mid 1 \leq i \leq n)$ *a mutually recursive equational system. Let* $(D, \sqsubseteq)$ *be given by* $D := (2^S)^n$ *and*

$$(T_1, \ldots, T_n) \sqsubseteq (T'_1, \ldots, T'_n) \qquad \textit{iff} \qquad T_i \subseteq T'_i \textit{ for every } i \in [n].$$

Then:

(1) $(D, \sqsubseteq)$ *is a complete lattice: if* $\{(T_1^k, \ldots, T_n^k) \mid k \in I\} \subseteq D$ *for some index set* I*, then*

$$\bigsqcup\{(T_1^k, \ldots, T_n^k) \mid k \in I\} = (\bigcup_{k \in I} T_1^k, \ldots, \bigcup_{k \in I} T_n^k)$$
$$\bigsqcap\{(T_1^k, \ldots, T_n^k) \mid k \in I\} = (\bigcap_{k \in I} T_1^k, \ldots, \bigcap_{k \in I} T_n^k)$$

Semantics of Recursive Equational Systems II

Lemma 11.6

Let $(S, Act, \longrightarrow)$ *be a finite LTS and* $E = (X_i = F_{X_i} \mid 1 \leq i \leq n)$ *a mutually recursive equational system. Let* $(D, \sqsubseteq)$ *be given by* $D := (2^S)^n$ *and*

$$(T_1, \ldots, T_n) \sqsubseteq (T'_1, \ldots, T'_n) \quad \text{iff} \quad T_i \subseteq T'_i \text{ for every } i \in [n].$$

Then:

(1) $(D, \sqsubseteq)$ *is a complete lattice: if* $\{(T_1^k, \ldots, T_n^k) \mid k \in I\} \subseteq D$ *for some index set* I*, then*

$$\bigsqcup\{(T_1^k, \ldots, T_n^k) \mid k \in I\} = \left(\bigcup_{k \in I} T_1^k, \ldots, \bigcup_{k \in I} T_n^k\right)$$
$$\bigsqcap\{(T_1^k, \ldots, T_n^k) \mid k \in I\} = \left(\bigcap_{k \in I} T_1^k, \ldots, \bigcap_{k \in I} T_n^k\right)$$

(2) $[\![E]\!]$ *is monotonic w.r.t.* $(D, \sqsubseteq)$

Semantics of Recursive Equational Systems II

Lemma 11.6

Let $(S, Act, \longrightarrow)$ *be a* *finite* *LTS and* $E = (X_i = F_{X_i} \mid 1 \leq i \leq n)$ *a mutually recursive equational system. Let* $(D, \sqsubseteq)$ *be given by* $D := (2^S)^n$ *and*

$$(T_1, \ldots, T_n) \sqsubseteq (T'_1, \ldots, T'_n) \quad \text{iff} \quad T_i \subseteq T'_i \text{ for every } i \in [n].$$

Then:

(1) $(D, \sqsubseteq)$ *is a complete lattice: if* $\{(T_1^k, \ldots, T_n^k) \mid k \in I\} \subseteq D$ *for some index set* I*, then*

$$\bigsqcup\{(T_1^k, \ldots, T_n^k) \mid k \in I\} = \left(\bigcup_{k \in I} T_1^k, \ldots, \bigcup_{k \in I} T_n^k\right)$$
$$\bigsqcap\{(T_1^k, \ldots, T_n^k) \mid k \in I\} = \left(\bigcap_{k \in I} T_1^k, \ldots, \bigcap_{k \in I} T_n^k\right)$$

(2) $[\![E]\!]$ *is monotonic w.r.t.* $(D, \sqsubseteq)$

(3) $\mathrm{lfp}([\![E]\!]) = [\![E]\!]^m(\emptyset, \ldots, \emptyset)$ *for some* $m \in \mathbb{N}$

(4) $\mathrm{gfp}([\![E]\!]) = [\![E]\!]^M(S, \ldots, S)$ *for some* $M \in \mathbb{N}$

Semantics of Recursive Equational Systems II

Lemma 11.6

Let $(S, Act, \longrightarrow)$ be a finite LTS and $E = (X_i = F_{X_i} \mid 1 \le i \le n)$ a mutually recursive equational system. Let $(D, \sqsubseteq)$ be given by $D := (2^S)^n$ and

$$(T_1, \ldots, T_n) \sqsubseteq (T'_1, \ldots, T'_n) \qquad \text{iff} \qquad T_i \subseteq T'_i \text{ for every } i \in [n].$$

Then:

(1) $(D, \sqsubseteq)$ *is a complete lattice: if* $\{(T_1^k, \ldots, T_n^k) \mid k \in I\} \subseteq D$ *for some index set* I, *then*

$$\bigsqcup\{(T_1^k, \ldots, T_n^k) \mid k \in I\} = \left(\bigcup_{k \in I} T_1^k, \ldots, \bigcup_{k \in I} T_n^k\right)$$
$$\bigsqcap\{(T_1^k, \ldots, T_n^k) \mid k \in I\} = \left(\bigcap_{k \in I} T_1^k, \ldots, \bigcap_{k \in I} T_n^k\right)$$

(2) $[\![E]\!]$ *is monotonic w.r.t.* $(D, \sqsubseteq)$

(3) $\mathrm{lfp}([\![E]\!]) = [\![E]\!]^m(\emptyset, \ldots, \emptyset)$ *for some* $m \in \mathbb{N}$

(4) $\mathrm{gfp}([\![E]\!]) = [\![E]\!]^M(S, \ldots, S)$ *for some* $M \in \mathbb{N}$

Proof.

omitted $\square$

A Mutually Recursive Specification

Example 11.7

$$
\begin{tikzcd}
s \arrow[r, bend left, "a"] & s_1 \arrow[l, bend left, "b"] & s_2 \arrow[l, "a"'] \arrow[r, "a"] & s_3 \arrow[loop right, "b"]
\end{tikzcd}
$$

- Let $S := \{s, s_1, s_2, s_3\}$ and E:

$$
\begin{aligned}
X &= \langle a\rangle \mathrm{tt} \wedge [a]Y \wedge [b]\mathrm{ff} \\
Y &= \langle b\rangle \mathrm{tt} \wedge [b]X \wedge [a]\mathrm{ff}
\end{aligned}
$$

Example 11.7

$$\begin{tikzcd} s \arrow[r, bend left, "a"] & s_1 \arrow[l, bend left, "b"] & s_2 \arrow[l, "a"'] \arrow[r, "a"] & s_3 \arrow[loop right, "b"] \end{tikzcd}$$

- Let $S := \{s, s_1, s_2, s_3\}$ and E:

$$X = \langle a \rangle \mathrm{tt} \wedge [a]Y \wedge [b]\mathrm{ff}$$
$$Y = \langle b \rangle \mathrm{tt} \wedge [b]X \wedge [a]\mathrm{ff}$$

- Interpretation:
  - X: “has no b-successor and ≥ 1 a-successors that all satisfy Y”
  - Y: “has no a-successor and ≥ 1 b-successors that all satisfy X”

A Mutually Recursive Specification

Example 11.7

$$\begin{tikzcd} s \arrow[r, bend left, "a"] & s_1 \arrow[l, bend left, "b"] & s_2 \arrow[l, "a"'] \arrow[r, "a"] & s_3 \arrow[loop right, "b"] \end{tikzcd}$$

- Let $S := \{s, s_1, s_2, s_3\}$ and E:

$$\begin{aligned} X &= \langle a \rangle \mathrm{tt} \wedge [a]Y \wedge [b]\mathrm{ff} \\ Y &= \langle b \rangle \mathrm{tt} \wedge [b]X \wedge [a]\mathrm{ff} \end{aligned}$$

- Interpretation:
  - X: “has no b-successor and ≥ 1 a-successors that all satisfy Y”
  - Y: “has no a-successor and ≥ 1 b-successors that all satisfy X”

  $\Rightarrow$ expected: $X = \{s\}$, $Y = \{s_1\}$

A Mutually Recursive Specification

Example 11.7

$$\begin{tikzcd} s \arrow[r, bend left, "a"] & s_1 \arrow[l, bend left, "b"] & s_2 \arrow[l, "a"'] \arrow[r, "a"] & s_3 \arrow[loop right, "b"] \end{tikzcd}$$

- Let $S := \{s, s_1, s_2, s_3\}$ and E:

$$\begin{aligned} X &= \langle a\rangle \mathrm{tt} \wedge [a]Y \wedge [b]\mathrm{ff} \\ Y &= \langle b\rangle \mathrm{tt} \wedge [b]X \wedge [a]\mathrm{ff} \end{aligned}$$

- Interpretation:
  - X: "has no b-successor and ≥ 1 a-successors that all satisfy Y"
  - Y: "has no a-successor and ≥ 1 b-successors that all satisfy X"

  $\Rightarrow$ expected: $X = \{s\}$, $Y = \{s_1\}$

- Formally: gfp of
  $[\![E]\!] : 2^S \times 2^S \to 2^S \times 2^S : (T_1, T_2) \mapsto$
  $(\{s, s_2\} \cap [\cdot a \cdot](T_2) \cap \{s, s_2\},$
  $\{s_1, s_3\} \cap [\cdot b \cdot](T_1) \cap \{s_1, s_3\})$

A Mutually Recursive Specification

Example 11.7

$$\begin{tikzcd} s \arrow[r, bend left, "a"] & s_1 \arrow[l, bend left, "b"] & s_2 \arrow[l, "a"'] \arrow[r, "a"] & s_3 \arrow[loop right, "b"] \end{tikzcd}$$

- Let $S := \{s, s_1, s_2, s_3\}$ and E:

$$X = \langle a \rangle \mathrm{tt} \wedge [a]Y \wedge [b]\mathrm{ff}$$
$$Y = \langle b \rangle \mathrm{tt} \wedge [b]X \wedge [a]\mathrm{ff}$$

- Interpretation:
  - X: “has no b-successor and ≥ 1 a-successors that all satisfy Y”
  - Y: “has no a-successor and ≥ 1 b-successors that all satisfy X”

  $\Rightarrow$ expected: $X = \{s\}$, $Y = \{s_1\}$
- Formally: gfp of $[\![E]\!] : 2^S \times 2^S \to 2^S \times 2^S : (T_1, T_2) \mapsto (\{s, s_2\} \cap [\cdot a \cdot](T_2) \cap \{s, s_2\}, \{s_1, s_3\} \cap [\cdot b \cdot](T_1) \cap \{s_1, s_3\})$

Fixed-point iteration:

$$\begin{aligned} &[\![E]\!]^1(S, S) \\ &= (\{s, s_2\} \cap S, \{s_1, s_3\} \cap S) \\ &= (\{s, s_2\}, \{s_1, s_3\}) \end{aligned}$$

A Mutually Recursive Specification

Example 11.7

$$\begin{tikzcd} s \arrow[r, bend left, "a"] & s_1 \arrow[l, bend left, "b"] & s_2 \arrow[l, "a"'] \arrow[r, "a"] & s_3 \arrow[loop right, "b"] \end{tikzcd}$$

- Let $S := \{s, s_1, s_2, s_3\}$ and E:

$$X = \langle a\rangle\mathrm{tt} \wedge [a]Y \wedge [b]\mathrm{ff}$$
$$Y = \langle b\rangle\mathrm{tt} \wedge [b]X \wedge [a]\mathrm{ff}$$

- Interpretation:
  - X: "has no b-successor and ≥ 1 a-successors that all satisfy Y"
  - Y: "has no a-successor and ≥ 1 b-successors that all satisfy X"

  $\Rightarrow$ expected: $X = \{s\}$, $Y = \{s_1\}$

- Formally: gfp of
  $[\![E]\!] : 2^S \times 2^S \to 2^S \times 2^S : (T_1, T_2) \mapsto (\{s, s_2\} \cap [\cdot a\cdot](T_2) \cap \{s, s_2\}, \{s_1, s_3\} \cap [\cdot b\cdot](T_1) \cap \{s_1, s_3\})$

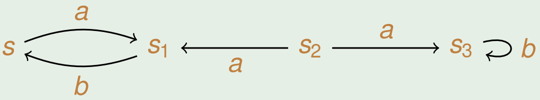
Fixed-point iteration:

$$[\![E]\!]^1(S, S) = (\{s, s_2\} \cap S, \{s_1, s_3\} \cap S) = (\{s, s_2\}, \{s_1, s_3\})$$

$$[\![E]\!]^2(S, S) = (\{s, s_2\} \cap S, \{s_1, s_3\} \cap \{s, s_1, s_2\}) = (\{s, s_2\}, \{s_1\})$$

A Mutually Recursive Specification

Example 11.7



- Let $S := \{s, s_1, s_2, s_3\}$ and E:

$$X = \langle a \rangle \mathrm{tt} \wedge [a]Y \wedge [b]\mathrm{ff}$$
$$Y = \langle b \rangle \mathrm{tt} \wedge [b]X \wedge [a]\mathrm{ff}$$

- Interpretation:
  - X: "has no b-successor and ≥ 1 a-successors that all satisfy Y"
  - Y: "has no a-successor and ≥ 1 b-successors that all satisfy X"

  $\Rightarrow$ expected: $X = \{s\}$, $Y = \{s_1\}$
- Formally: gfp of
  $[\![E]\!] : 2^S \times 2^S \to 2^S \times 2^S : (T_1, T_2) \mapsto$
  $(\{s, s_2\} \cap [\cdot a \cdot](T_2) \cap \{s, s_2\},$
  $\{s_1, s_3\} \cap [\cdot b \cdot](T_1) \cap \{s_1, s_3\})$

Fixed-point iteration:

$$[\![E]\!]^1(S, S)$$
$$= (\{s, s_2\} \cap S, \{s_1, s_3\} \cap S)$$
$$= (\{s, s_2\}, \{s_1, s_3\})$$

$$[\![E]\!]^2(S, S)$$
$$= (\{s, s_2\} \cap S, \{s_1, s_3\} \cap \{s, s_1, s_2\})$$
$$= (\{s, s_2\}, \{s_1\})$$

$$[\![E]\!]^3(S, S)$$
$$= (\{s, s_2\} \cap \{s, s_1, s_3\}, \{s_1, s_3\} \cap \{s, s_1, s_2\}$$
$$= (\{s\}, \{s_1\})$$

A Mutually Recursive Specification

Example 11.7

$$
\begin{tikzcd}
s \arrow[r, bend left, "a"] & s_1 \arrow[l, bend left, "b"] & s_2 \arrow[l, "a"'] \arrow[r, "a"] & s_3 \arrow[loop right, "b"]
\end{tikzcd}
$$

- Let $S := \{s, s_1, s_2, s_3\}$ and E:

$$
\begin{aligned}
X &= \langle a\rangle \mathrm{tt} \wedge [a]Y \wedge [b]\mathrm{ff}\\
Y &= \langle b\rangle \mathrm{tt} \wedge [b]X \wedge [a]\mathrm{ff}
\end{aligned}
$$

- Interpretation:
  - X: "has no b-successor and ≥ 1 a-successors that all satisfy Y"
  - Y: "has no a-successor and ≥ 1 b-successors that all satisfy X"

  $\Rightarrow$ expected: $X = \{s\}$, $Y = \{s_1\}$

- Formally: gfp of
  $[\![E]\!] : 2^S \times 2^S \to 2^S \times 2^S : (T_1, T_2) \mapsto$
  $(\{s, s_2\} \cap [\cdot a \cdot](T_2) \cap \{s, s_2\},$
  $\{s_1, s_3\} \cap [\cdot b \cdot](T_1) \cap \{s_1, s_3\})$

Fixed-point iteration:

$$
\begin{aligned}
&[\![E]\!]^1(S, S)\\
&= (\{s, s_2\} \cap S, \{s_1, s_3\} \cap S)\\
&= (\{s, s_2\}, \{s_1, s_3\})
\end{aligned}
$$

$$
\begin{aligned}
&[\![E]\!]^2(S, S)\\
&= (\{s, s_2\} \cap S, \{s_1, s_3\} \cap \{s, s_1, s_2\})\\
&= (\{s, s_2\}, \{s_1\})
\end{aligned}
$$

$$
\begin{aligned}
&[\![E]\!]^3(S, S)\\
&= (\{s, s_2\} \cap \{s, s_1, s_3\}, \{s_1, s_3\} \cap \{s, s_1, s_2\}\\
&= (\{s\}, \{s_1\})
\end{aligned}
$$

$$
\begin{aligned}
&[\![E]\!]^4(S, S)\\
&= [\![E]\!]^3(S, S)
\end{aligned}
$$

$$\Rightarrow \mathrm{gfp}([\![E]\!]) = (\{s\}, \{s_1\})$$

Outline of Lecture 11

- The Hennessy-Milner theorem asserts that for finitely branching processes, strong bisimilarity and HML-equivalence coincide.

Characteristic Formulae

- The Hennessy-Milner theorem asserts that for finitely branching processes, strong bisimilarity and HML-equivalence coincide.
- As a next step, we show that for finite transition systems, the equivalence classes under $\sim$ can be characterised by a system of formulae in HML extended with recursion – one for each state.

Characteristic Formulae

- The Hennessy-Milner theorem asserts that for finitely branching processes, strong bisimilarity and HML-equivalence coincide.
- As a next step, we show that for finite transition systems, the equivalence classes under $\sim$ can be characterised by a system of formulae in HML extended with recursion – one for each state.
- For a finite process P, this HML-formula is called P's characteristic formula as it exactly characterises the $\sim$-equivalence class of P.

Lemma 11.8

There is no recursion-free formula $F \in HMF$ that can characterise the process $A^\omega = a.A^\omega$ up to strong bisimilarity.

The Need for Recursion

Lemma 11.8

There is no recursion-free formula $F \in HMF$ *that can characterise the process* $A^\omega = a.A^\omega$ *up to strong bisimilarity.*

Proof.

- Assume $F \in HMF$ with $[\![F]\!] = \{P \in Prc \mid P \sim A^\omega\}$.

$\square$

The Need for Recursion

Lemma 11.8

There is no recursion-free formula $F \in HMF$ that can characterise the process $A^\omega = a.A^\omega$ up to strong bisimilarity.

Proof.

- Assume $F \in HMF$ with $[\![F]\!] = \{P \in Prc \mid P \sim A^\omega\}$.
- Obviously $a^i \not\sim A^\omega$ for every $i \geq 0$.

$\square$

The Need for Recursion

Lemma 11.8

There is no recursion-free formula $F \in HMF$ that can characterise the process $A^\omega = a.A^\omega$ up to strong bisimilarity.

Proof.

- Assume $F \in HMF$ with $[\![F]\!] = \{P \in Prc \mid P \sim A^\omega\}$.
- Obviously $a^i \not\sim A^\omega$ for every $i \geq 0$.
- On the other hand, $A^\omega \models F$ implies (by Lemma 9.9) that $a^k \models F$ where k is the modal depth of F.

$\square$

Lemma (Lemma 9.9)

For every $F \in HMF$, $A^\omega \models F$ iff $a^k \models F$, where k is the modal depth[a] of F.

[a] the maximal number of nested occurrences of modal operators in F

The Need for Recursion

Lemma 11.8

There is no recursion-free formula $F \in HMF$ that can characterise the process $A^\omega = a.A^\omega$ up to strong bisimilarity.

Proof.

- Assume $F \in HMF$ with $[\![F]\!] = \{P \in Prc \mid P \sim A^\omega\}$.
- Obviously $a^i \not\sim A^\omega$ for every $i \geq 0$.
- On the other hand, $A^\omega \models F$ implies (by Lemma 9.9) that $a^k \models F$ where k is the modal depth of F.
- Thus, $a^k \sim A^\omega$, which contradicts $a^i \not\sim A^\omega$. □

Lemma (Lemma 9.9)

For every $F \in HMF$, $A^\omega \models F$ iff $a^k \models F$, where k is the modal depth[a] of F.

[a] the maximal number of nested occurrences of modal operators in F

Characteristic Formulae by Example

- Consider the finite LTS $(S, Act, \longrightarrow)$, and let $\mathcal{X}$ contain (at least) $|S|$ variables.

Characteristic Formulae by Example

- Consider the finite LTS $(S, Act, \longrightarrow)$, and let $\mathcal{X}$ contain (at least) $|S|$ variables.
- Intuitively, X_P is the syntactic symbol for the characteristic formula of process $P \in S$.

Characteristic Formulae by Example

- Consider the finite LTS $(S, Act, \longrightarrow)$, and let $\mathcal{X}$ contain (at least) $|S|$ variables.
- Intuitively, X_P is the syntactic symbol for the characteristic formula of process $P \in S$.
- A characteristic formula for P has to describe
  - which actions P can perform,
  - what happens after performing an action, and
  - which actions it cannot perform.

Characteristic Formulae by Example

- Consider the finite LTS $(S, Act, \longrightarrow)$, and let $\mathcal{X}$ contain (at least) $|S|$ variables.
- Intuitively, X_P is the syntactic symbol for the characteristic formula of process $P \in S$.
- A characteristic formula for P has to describe
  - which actions P can perform,
  - what happens after performing an action, and
  - which actions it cannot perform.

Example 11.9 (Coffee/tea machine; cf. Example 3.13)

$$M = m.M' \qquad M' = \overline{c}.M + \overline{t}.M$$

Characteristic Formulae by Example

- Consider the finite LTS $(S, Act, \longrightarrow)$, and let $\mathcal{X}$ contain (at least) $|S|$ variables.
- Intuitively, X_P is the syntactic symbol for the characteristic formula of process $P \in S$.
- A characteristic formula for P has to describe
  - which actions P can perform,
  - what happens after performing an action, and
  - which actions it cannot perform.

Example 11.9 (Coffee/tea machine; cf. Example 3.13)

$$M = m.M' \qquad M' = \overline{c}.M + \overline{t}.M$$

Observations:

(1) M can perform m and become M'

(2) Performing m, M necessarily becomes M'

(3) M cannot perform any other action than m

Characteristic Formulae by Example

- Consider the finite LTS $(S, Act, \longrightarrow)$, and let $\mathcal{X}$ contain (at least) $|S|$ variables.
- Intuitively, X_P is the syntactic symbol for the characteristic formula of process $P \in S$.
- A characteristic formula for P has to describe
  - which actions P can perform,
  - what happens after performing an action, and
  - which actions it cannot perform.

Example 11.9 (Coffee/tea machine; cf. Example 3.13)

$$M = m.M' \qquad M' = \overline{c}.M + \overline{t}.M$$

Observations:

(1) M can perform m and become M'

(2) Performing m, M necessarily becomes M'

(3) M cannot perform any other action than m

Therefore:

$$X_M = \langle m \rangle X_{M'} \wedge [m] X_{M'} \wedge [\{\overline{c}, \overline{t}\}]\mathrm{ff}$$

Characteristic Formulae by Example

- Consider the finite LTS $(S, Act, \longrightarrow)$, and let $\mathcal{X}$ contain (at least) $|S|$ variables.
- Intuitively, X_P is the syntactic symbol for the characteristic formula of process $P \in S$.
- A characteristic formula for P has to describe
  - which actions P can perform,
  - what happens after performing an action, and
  - which actions it cannot perform.

Example 11.9 (Coffee/tea machine; cf. Example 3.13)

$$M = m.M' \qquad M' = \overline{c}.M + \overline{t}.M$$

Observations:

(1) M can perform m and become M'

(2) Performing m, M necessarily becomes M'

(3) M cannot perform any other action than m

Therefore:

$$X_M = \langle m\rangle X_{M'} \wedge [m]X_{M'} \wedge [\{\overline{c}, \overline{t}\}]\mathrm{ff}$$

Similarly:

$$X_{M'} = \langle\overline{c}\rangle X_M \wedge \langle\overline{t}\rangle X_M \wedge [\{\overline{c}, \overline{t}\}]X_M \wedge [m]\mathrm{ff}$$

The General Case

Enabled actions: $P \models \bigwedge_{\{\alpha, P' \mid P \xrightarrow{\alpha} P'\}} \langle\alpha\rangle X_{P'}$

Resulting states: $P \models \bigwedge_{\{\alpha \mid P \xrightarrow{\alpha}\}} [\alpha] \left(\bigvee_{\{P' \mid P \xrightarrow{\alpha} P'\}} X_{P'} \right)$

Disabled actions: $P \models \bigwedge_{\{\alpha \mid P \not\xrightarrow{\alpha}\}} [\alpha]\mathrm{ff}$

The General Case

Enabled actions: $P \models \bigwedge_{\{\alpha,P' \mid P \xrightarrow{\alpha} P'\}} \langle\alpha\rangle X_{P'}$

Resulting states: $P \models \bigwedge_{\{\alpha \mid P \xrightarrow{\alpha}\}} [\alpha] \left(\bigvee_{\{P' \mid P \xrightarrow{\alpha} P'\}} X_{P'}\right)$

Disabled actions: $P \models \bigwedge_{\{\alpha \mid P \not\xrightarrow{\alpha}\}} [\alpha]\mathrm{ff}$

can be combined!

Enabled actions: $P \models \bigwedge_{\{\alpha,P' \mid P \xrightarrow{\alpha} P'\}} \langle\alpha\rangle X_{P'}$

Resulting states: $P \models \bigwedge_{\{\alpha \mid P \xrightarrow{\alpha}\}} [\alpha] \left(\bigvee_{\{P' \mid P \xrightarrow{\alpha} P'\}} X_{P'} \right)$

Disabled actions: $P \models \bigwedge_{\{\alpha \mid P \not\xrightarrow{\alpha}\}} [\alpha]\mathrm{ff}$

can be combined!

Theorem 11.10 (Characteristic Formula) (Ingolfsdottir et al. 1987)

For a finite-state process $P \in Prc$, let the characteristic formula $X_P \in HMF_{\mathcal{X}}$ be defined by:

$$X_P \stackrel{max}{=} \bigwedge_{\{\alpha,P' \mid P \xrightarrow{\alpha} P'\}} \langle\alpha\rangle X_{P'} \wedge \bigwedge_{\alpha \in Act} [\alpha] \left(\bigvee_{\{P' \mid P \xrightarrow{\alpha} P'\}} X_{P'} \right)$$

(where $\bigwedge_\emptyset \ldots := \mathrm{tt}$ and $\bigvee_\emptyset \ldots := \mathrm{ff}$). Then, for every $Q \in Prc$: $Q \models X_P$ iff $P \sim Q$.

Proof.

omitted □

Outline of Lecture 11

Mixing Least and Greatest Fixed Points I

- **So far:** least/greatest fixed point of overall system
- **But:** too restrictive

Mixing Least and Greatest Fixed Points I

- **So far:** least/greatest fixed point of overall system
- **But:** too restrictive

Example 11.11

"It is possible for the system to reach a state which has a livelock (i.e., an outgoing infinite sequence of internal steps)."

Mixing Least and Greatest Fixed Points I

- **So far:** least/greatest fixed point of overall system
- **But:** too restrictive

Example 11.11

> *"It is possible for the system to reach a state which has a livelock (i.e., an outgoing infinite sequence of internal steps)."*

can be specified by

$$Pos(Livelock)$$

where

$$\begin{aligned} Pos(F) &\overset{min}{=} F \vee \langle Act \rangle Pos(F) && \text{(cf. Theorem 11.2)} \\ Livelock &\overset{max}{=} \langle \tau \rangle Livelock \end{aligned}$$

and thus $Livelock \equiv Forever(\tau)$ (cf. Example 11.3).

Mixing Least and Greatest Fixed Points II

Caveat: Arbitrary mixing can entail non-monotonic behaviour!

Mixing Least and Greatest Fixed Points II

Caveat: Arbitrary mixing can entail non-monotonic behaviour!

Example 11.12

$$
\begin{aligned}
E : X &\overset{min}{=} Y \\
Y &\overset{max}{=} X
\end{aligned}
$$

Mixing Least and Greatest Fixed Points II

Caveat: Arbitrary mixing can entail non-monotonic behaviour!

Example 11.12

$$E : X \overset{min}{=} Y$$
$$Y \overset{max}{=} X$$

Fixed-point iteration:

$$(\perp, \top) = (\emptyset, S)$$

Mixing Least and Greatest Fixed Points II

Caveat: Arbitrary mixing can entail non-monotonic behaviour!

Example 11.12

$$E: X \stackrel{min}{=} Y$$
$$Y \stackrel{max}{=} X$$

Fixed-point iteration:

$$(\bot, \top) = (\emptyset, S) \stackrel{[\![E]\!]}{\mapsto} (S, \emptyset)$$

Mixing Least and Greatest Fixed Points II

Caveat: Arbitrary mixing can entail non-monotonic behaviour!

Example 11.12

$$E : X \overset{min}{=} Y$$
$$Y \overset{max}{=} X$$

Fixed-point iteration:

$$(\bot, \top) = (\emptyset, S) \overset{[\![E]\!]}{\mapsto} (S, \emptyset) \overset{[\![E]\!]}{\mapsto} (\emptyset, S) \overset{[\![E]\!]}{\mapsto} \ldots$$

Mixing Least and Greatest Fixed Points II

Caveat: Arbitrary mixing can entail non-monotonic behaviour!

Example 11.12

$$\begin{aligned} E : X &\stackrel{min}{=} Y \\ Y &\stackrel{max}{=} X \end{aligned}$$

Fixed-point iteration:

$$(\bot, \top) = (\emptyset, S) \stackrel{[\![E]\!]}{\mapsto} (S, \emptyset) \stackrel{[\![E]\!]}{\mapsto} (\emptyset, S) \stackrel{[\![E]\!]}{\mapsto} \ldots$$

Solution: Nesting of specifications by partitioning equations into a sequence of blocks such that all equations in one block

- are of same type (either *min* or *max*) and
- use only variables defined in the same or subsequent blocks.

Mixing Least and Greatest Fixed Points II

Caveat: Arbitrary mixing can entail non-monotonic behaviour!

Example 11.12

$$E : X \stackrel{min}{=} Y$$
$$Y \stackrel{max}{=} X$$

Fixed-point iteration:

$$(\bot, \top) = (\emptyset, S) \stackrel{[\![E]\!]}{\mapsto} (S, \emptyset) \stackrel{[\![E]\!]}{\mapsto} (\emptyset, S) \stackrel{[\![E]\!]}{\mapsto} \ldots$$

Solution: Nesting of specifications by partitioning equations into a sequence of blocks such that all equations in one block

- are of same type (either *min* or *max*) and
- use only variables defined in the same or subsequent blocks.

$\Rightarrow$ Bottom-up, block-wise evaluation by fixed-point iteration

Mixing Least and Greatest Fixed Points III

Example 11.13 (cf. Example 11.11)

$$PosLL \overset{min}{=} Livelock \vee \langle Act \rangle PosLL$$
$$Livelock \overset{max}{=} \langle \tau \rangle Livelock$$

$$\begin{tikzcd} s \arrow[r, "a"] & p \arrow[r, "\tau"] \arrow[loop below, "\tau"] & q \arrow[r, "\tau"] & r \end{tikzcd}$$

Example 11.13 (cf. Example 11.11)

$$PosLL \overset{min}{=} Livelock \vee \langle Act \rangle PosLL$$
$$Livelock \overset{max}{=} \langle \tau \rangle Livelock$$

$$s \xrightarrow{a} p \xrightarrow{\tau} q \xrightarrow{\tau} r$$

$$p \circlearrowleft \tau$$

(1) Greatest fixed-point iteration for *Livelock* $: T \mapsto \langle \cdot \tau \cdot \rangle (T)$:

$$\top = S = \{s, p, q, r\}$$

Mixing Least and Greatest Fixed Points III

Example 11.13 (cf. Example 11.11)

$$PosLL \overset{min}{=} Livelock \vee \langle Act \rangle PosLL$$
$$Livelock \overset{max}{=} \langle \tau \rangle Livelock$$

$$\begin{tikzcd} s \arrow[r, "a"] & p \arrow[r, "\tau"] \arrow[loop below, "\tau"] & q \arrow[r, "\tau"] & r \end{tikzcd}$$

(1) Greatest fixed-point iteration for *Livelock* : $T \mapsto \langle \cdot \tau \cdot \rangle (T)$:

$$\top = S = \{s, p, q, r\} \mapsto \{p, q\}$$

Example 11.13 (cf. Example 11.11)

$$PosLL \overset{min}{=} Livelock \vee \langle Act \rangle PosLL$$
$$Livelock \overset{max}{=} \langle \tau \rangle Livelock$$

$$\begin{tikzcd} s \arrow[r, "a"] & p \arrow[r, "\tau"] \arrow[loop below, "\tau"] & q \arrow[r, "\tau"] & r \end{tikzcd}$$

(1) Greatest fixed-point iteration for $Livelock : T \mapsto \langle \cdot \tau \cdot \rangle (T)$:

$$\top = S = \{s, p, q, r\} \mapsto \{p, q\} \mapsto \{p\}$$

Mixing Least and Greatest Fixed Points III

Example 11.13 (cf. Example 11.11)

$$PosLL \stackrel{min}{=} Livelock \vee \langle Act \rangle PosLL$$
$$Livelock \stackrel{max}{=} \langle \tau \rangle Livelock$$

$$\begin{tikzcd} s \arrow[r, "a"] & p \arrow[r, "\tau"] \arrow[loop below, "\tau"] & q \arrow[r, "\tau"] & r \end{tikzcd}$$

(1) Greatest fixed-point iteration for $Livelock : T \mapsto \langle \cdot \tau \cdot \rangle (T)$:

$$\top = S = \{s, p, q, r\} \mapsto \{p, q\} \mapsto \{p\} \mapsto \{p\}$$

Mixing Least and Greatest Fixed Points III

Example 11.13 (cf. Example 11.11)

$$PosLL \overset{min}{=} Livelock \vee \langle Act \rangle PosLL$$
$$Livelock \overset{max}{=} \langle \tau \rangle Livelock$$

$$\begin{tikzcd} s \arrow[r, "a"] & p \arrow[r, "\tau"] \arrow[loop below, "\tau"] & q \arrow[r, "\tau"] & r \end{tikzcd}$$

(1) Greatest fixed-point iteration for $Livelock : T \mapsto \langle \cdot \tau \cdot \rangle (T)$:

$$\top = S = \{s, p, q, r\} \mapsto \{p, q\} \mapsto \{p\} \mapsto \{p\}$$

(2) Least fixed-point iteration for $PosLL : T \mapsto \{p\} \cup \langle \cdot Act \cdot \rangle (T)$:

$$\bot = \emptyset$$

Example 11.13 (cf. Example 11.11)

$$PosLL \overset{min}{=} Livelock \vee \langle Act \rangle PosLL$$
$$Livelock \overset{max}{=} \langle \tau \rangle Livelock$$

$$\begin{tikzcd} s \arrow[r, "a"] & p \arrow[r, "\tau"] \arrow[loop below, "\tau"] & q \arrow[r, "\tau"] & r \end{tikzcd}$$

(1) Greatest fixed-point iteration for $Livelock : T \mapsto \langle \cdot \tau \cdot \rangle (T)$:

$$\top = S = \{s, p, q, r\} \mapsto \{p, q\} \mapsto \{p\} \mapsto \{p\}$$

(2) Least fixed-point iteration for $PosLL : T \mapsto \{p\} \cup \langle \cdot Act \cdot \rangle (T)$:

$$\bot = \emptyset \mapsto \{p\}$$

Mixing Least and Greatest Fixed Points III

Example 11.13 (cf. Example 11.11)

$$PosLL \overset{min}{=} Livelock \vee \langle Act \rangle PosLL$$
$$Livelock \overset{max}{=} \langle \tau \rangle Livelock$$

$$\begin{tikzcd} s \arrow[r, "a"] & p \arrow[r, "\tau"] \arrow[loop below, "\tau"] & q \arrow[r, "\tau"] & r \end{tikzcd}$$

(1) Greatest fixed-point iteration for $Livelock : T \mapsto \langle \cdot\tau\cdot \rangle (T)$:

$$\top = S = \{s, p, q, r\} \mapsto \{p, q\} \mapsto \{p\} \mapsto \{p\}$$

(2) Least fixed-point iteration for $PosLL : T \mapsto \{p\} \cup \langle \cdot Act \cdot \rangle (T)$:

$$\bot = \emptyset \mapsto \{p\} \mapsto \{s, p\}$$

Mixing Least and Greatest Fixed Points III

Example 11.13 (cf. Example 11.11)

$$PosLL \overset{min}{=} Livelock \vee \langle Act \rangle PosLL$$
$$Livelock \overset{max}{=} \langle \tau \rangle Livelock$$

$$\begin{tikzcd} s \arrow[r, "a"] & p \arrow[r, "\tau"] \arrow[loop below, "\tau"] & q \arrow[r, "\tau"] & r \end{tikzcd}$$

(1) Greatest fixed-point iteration for $Livelock : T \mapsto \langle \cdot \tau \cdot \rangle (T)$:

$$\top = S = \{s, p, q, r\} \mapsto \{p, q\} \mapsto \{p\} \mapsto \{p\}$$

(2) Least fixed-point iteration for $PosLL : T \mapsto \{p\} \cup \langle \cdot Act \cdot \rangle (T)$:

$$\bot = \emptyset \mapsto \{p\} \mapsto \{s, p\} \mapsto \{s, p\}$$

The Modal μ-Calculus

- Logic that supports free mixing of least and greatest fixed points (with guardedness conditions):
  - originally introduced by D. Kozen: *Results on the Propositional μ-Calculus*, Theoretical Computer Science 27, 1983
  - overview paper by J. Bradfield, C. Stirling: *Modal Logics and mu-Calculi: An Introduction*, Chapter 4 of *Handbook of Process Algebra*, Elsevier, 2001

The Modal μ-Calculus

- Logic that supports free mixing of least and greatest fixed points (with guardedness conditions):
  - originally introduced by D. Kozen: *Results on the Propositional μ-Calculus*, Theoretical Computer Science 27, 1983
  - overview paper by J. Bradfield, C. Stirling: *Modal Logics and mu-Calculi: An Introduction*, Chapter 4 of *Handbook of Process Algebra*, Elsevier, 2001
- HML variants are fragments thereof
- Expressivity increases with alternation of least and greatest fixed points:
  - J.C. Bradfield: *The Modal Mu-Calculus Alternation Hierarchy is Strict*, Theoretical Computer Science 195(2), 1998

The Modal μ-Calculus

- Logic that supports free mixing of least and greatest fixed points (with guardedness conditions):
  - originally introduced by D. Kozen: *Results on the Propositional μ-Calculus*, Theoretical Computer Science 27, 1983
  - overview paper by J. Bradfield, C. Stirling: *Modal Logics and mu-Calculi: An Introduction*, Chapter 4 of *Handbook of Process Algebra*, Elsevier, 2001
- HML variants are fragments thereof
- Expressivity increases with alternation of least and greatest fixed points:
  - J.C. Bradfield: *The Modal Mu-Calculus Alternation Hierarchy is Strict*, Theoretical Computer Science 195(2), 1998
- Decidable model-checking problem for finite LTSs
  (in NP $\cap$ co-NP; linear for HML with one variable)
- Generally undecidable for infinite LTSs and HML with one variable (CCS, Petri nets, ...)
- Overview paper:
  - O. Burkart, D. Caucal, F. Moller, B. Steffen: *Verification on Infinite Structures*, Chapter 9 of *Handbook of Process Algebra*, Elsevier, 2001