Concurrency Theory

Winter Semester 2017/18

Lecture 6: Mutually Recursive Equational Systems

Joost-Pieter Katoen and Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

http://moves.rwth-aachen.de/teaching/ws-1718/ct/

Software Modeling and Verification Chair | RWTH AACHEN UNIVERSITY

Outline of Lecture 6

Recap: Fixed-Point Theory

Largest Fixed Points and Invariants

Mutually Recursive Equational Systems

Mixing Least and Greatest Fixed Points

The Fixed-Point Theorem I

Alfred Tarski (1901–1983)

Theorem (Tarski's fixed-point theorem)

Let $(D, \sqsubseteq)$ *be a complete lattice and* $f : D \to D$ *monotonic. Then* f *has a least fixed point* $\text{fix}(f)$ *and a greatest fixed point* $\text{FIX}(f)$ *given by*

$$\text{fix}(f) = \bigsqcap\{d \in D \mid f(d) \sqsubseteq d\} \qquad \textit{(GLB of all pre-fixed points of } f\textit{)}$$

$$\text{FIX}(f) = \bigsqcup\{d \in D \mid d \sqsubseteq f(d)\} \qquad \textit{(LUB of all post-fixed points of } f\textit{)}$$

Proof.

on the board □

The Fixed-Point Theorem for Finite Lattices

Theorem (Fixed-point theorem for finite lattices)

Let $(D, \sqsubseteq)$ *be a finite complete lattice and* $f : D \to D$ *monotonic. Then*

$$\mathrm{fix}(f) = f^m(\bot) \quad \textit{and} \quad \mathrm{FIX}(f) = f^M(\top)$$

for some $m, M \in \mathbb{N}$ *where* $f^0(d) := d \quad \textit{and} \quad f^{k+1}(d) := f(f^k(d))$.

Proof.

on the board □

Application to HML with Recursion

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. $\mathrm{fix}([\![F]\!]) = \bigcap\{T \subseteq S \mid [\![F]\!](T) \subseteq T\}$
3. $\mathrm{FIX}([\![F]\!]) = \bigcup\{T \subseteq S \mid T \subseteq [\![F]\!](T)\}$

If, in addition, S is finite, then

4. $\mathrm{fix}([\![F]\!]) = [\![F]\!]^m(\emptyset)$ *for some* $m \in \mathbb{N}$
5. $\mathrm{FIX}([\![F]\!]) = [\![F]\!]^M(S)$ *for some* $M \in \mathbb{N}$

Proof.

1. by induction on the structure of F (details omitted)
2. by Lemma 4.15 and Theorem 5.5
3. by Lemma 4.15 and Theorem 5.5
4. by Lemma 4.15 and Theorem 5.7
5. by Lemma 4.15 and Theorem 5.7 □

Outline of Lecture 6

Largest Fixed Points and Invariants

- Remember (Example 4.5):
  - Invariant: $Inv(F) \equiv X$ for $F \in HMF$ and $X \stackrel{max}{=} F \wedge [Act]X$
  - $s \models Inv(F)$ if all states reachable from s satisfy F

Largest Fixed Points and Invariants

- Remember (Example 4.5):
  - Invariant: $Inv(F) \equiv X$ for $F \in HMF$ and $X \stackrel{max}{=} F \wedge [Act]X$
  - $s \models Inv(F)$ if all states reachable from s satisfy F
- Now: formalize argument and prove its correctness (for arbitrary LTSs)

Largest Fixed Points and Invariants

- Remember (Example 4.5):
  - Invariant: $Inv(F) \equiv X$ for $F \in HMF$ and $X \stackrel{max}{=} F \wedge [Act]X$
  - $s \models Inv(F)$ if all states reachable from s satisfy F
- Now: formalize 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 Theorem 5.5, $\text{FIX}(inv) = \bigcup \{T \subseteq S \mid T \subseteq inv(T)\}$

Largest Fixed Points and Invariants

- Remember (Example 4.5):
  - Invariant: $Inv(F) \equiv X$ for $F \in HMF$ and $X \stackrel{max}{=} F \wedge [Act]X$
  - $s \models Inv(F)$ if all states reachable from s satisfy F
- Now: formalize 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 Theorem 5.5, $\text{FIX}(inv) = \bigcup\{T \subseteq S \mid T \subseteq inv(T)\}$
- Direct formulation of invariance property:

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

Largest Fixed Points and Invariants

- Remember (Example 4.5):
  - Invariant: $Inv(F) \equiv X$ for $F \in HMF$ and $X \overset{max}{=} F \wedge [Act]X$
  - $s \models Inv(F)$ if all states reachable from s satisfy F
- Now: formalize 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 Theorem 5.5, $\text{FIX}(inv) = \bigcup\{T \subseteq S \mid T \subseteq inv(T)\}$
- Direct formulation of invariance property:

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

Theorem 6.1

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

Largest Fixed Points and Invariants

- Remember (Example 4.5):
  - Invariant: $Inv(F) \equiv X$ for $F \in HMF$ and $X \overset{max}{=} F \land [Act]X$
  - $s \models Inv(F)$ if all states reachable from s satisfy F
- Now: formalize 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 Theorem 5.5, $\text{FIX}(inv) = \bigcup\{T \subseteq S \mid T \subseteq inv(T)\}$
- Direct formulation of invariance property:

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

Theorem 6.1

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

Proof.

on the board □

Outline of Lecture 6

Introducing Several Variables

Sometimes useful: using more than one variable

Example 6.2

"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 useful: using more than one variable

Example 6.2

> *"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

$$Inv(F) \stackrel{max}{=} F \wedge [Act]F \qquad \text{(cf. Theorem 6.1)}$$
$$Forever(b) \stackrel{max}{=} \langle b \rangle Forever(b)$$

Syntax of Mutually Recursive Equational Systems

Definition 6.3 (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:

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

where $1 \leq i \leq n$ 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 $1 \leq i \leq n$.

Semantics of Recursive Equational Systems I

As before: semantics of formula depends on states satisfying the variables

Definition 6.4 (Semantics of mutually recursive equational systems)

Let $(S, Act, \longrightarrow)$ be an LTS and $E = (X_i = F_{X_i} \mid 1 \leq i \leq n)$ 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

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

Semantics of Recursive Equational Systems II

Lemma 6.5

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*

$$(T_1, \ldots, T_n) \sqsubseteq (T'_1, \ldots, T'_n)$$

iff $T_i \subseteq T'_i$ *for every* $1 \leq i \leq n$.

Semantics of Recursive Equational Systems II

Lemma 6.5

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*

$$(T_1, \dots, T_n) \sqsubseteq (T'_1, \dots, T'_n)$$

iff $T_i \subseteq T'_i$ *for every* $1 \leq i \leq n$*.*

1. $(D, \sqsubseteq)$ *is a complete lattice with*

$$\bigsqcup\{(T^i_1, \dots, T^i_n) \mid i \in I\} = \left(\bigcup\{T^i_1 \mid i \in I\}, \dots, \bigcup\{T^i_n \mid i \in I\}\right)$$
$$\bigsqcap\{(T^i_1, \dots, T^i_n) \mid i \in I\} = \left(\bigcap\{T^i_1 \mid i \in I\}, \dots, \bigcap\{T^i_n \mid i \in I\}\right)$$

Semantics of Recursive Equational Systems II

Lemma 6.5

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*

$$(T_1, \ldots, T_n) \sqsubseteq (T'_1, \ldots, T'_n)$$

iff $T_i \subseteq T'_i$ *for every* $1 \leq i \leq n$*.*

1. $(D, \sqsubseteq)$ *is a complete lattice with*

$$\bigsqcup\{(T_1^i, \ldots, T_n^i) \mid i \in I\} = \left(\bigcup\{T_1^i \mid i \in I\}, \ldots, \bigcup\{T_n^i \mid i \in I\}\right)$$
$$\bigsqcap\{(T_1^i, \ldots, T_n^i) \mid i \in I\} = \left(\bigcap\{T_1^i \mid i \in I\}, \ldots, \bigcap\{T_n^i \mid i \in I\}\right)$$

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

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Semantics of Recursive Equational Systems II

Lemma 6.5

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*

$$(T_1, \ldots, T_n) \sqsubseteq (T'_1, \ldots, T'_n)$$

iff $T_i \subseteq T'_i$ *for every* $1 \leq i \leq n$.

1. $(D, \sqsubseteq)$ *is a complete lattice with*

$$\bigsqcup\{(T^i_1, \ldots, T^i_n) \mid i \in I\} = \left(\bigcup\{T^i_1 \mid i \in I\}, \ldots, \bigcup\{T^i_n \mid i \in I\}\right)$$
$$\bigsqcap\{(T^i_1, \ldots, T^i_n) \mid i \in I\} = \left(\bigcap\{T^i_1 \mid i \in I\}, \ldots, \bigcap\{T^i_n \mid i \in I\}\right)$$

2. $[\![E]\!]$ *is monotonic w.r.t.* $(D, \sqsubseteq)$
3. $\text{fix}([\![E]\!]) = [\![E]\!]^m(\emptyset, \ldots, \emptyset)$ *for some* $m \in \mathbb{N}$
4. $\text{FIX}([\![E]\!]) = [\![E]\!]^M(S, \ldots, S)$ *for some* $M \in \mathbb{N}$

Semantics of Recursive Equational Systems II

Lemma 6.5

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*

$$(T_1, \ldots, T_n) \sqsubseteq (T'_1, \ldots, T'_n)$$

iff $T_i \subseteq T'_i$ *for every* $1 \leq i \leq n$.

1. $(D, \sqsubseteq)$ *is a complete lattice with*
$$\bigsqcup\{(T^i_1, \ldots, T^i_n) \mid i \in I\} = \left(\bigcup\{T^i_1 \mid i \in I\}, \ldots, \bigcup\{T^i_n \mid i \in I\}\right)$$
$$\bigsqcap\{(T^i_1, \ldots, T^i_n) \mid i \in I\} = \left(\bigcap\{T^i_1 \mid i \in I\}, \ldots, \bigcap\{T^i_n \mid i \in I\}\right)$$
2. $[\![E]\!]$ *is monotonic w.r.t.* $(D, \sqsubseteq)$
3. $\mathrm{fix}([\![E]\!]) = [\![E]\!]^m(\emptyset, \ldots, \emptyset)$ *for some* $m \in \mathbb{N}$
4. $\mathrm{FIX}([\![E]\!]) = [\![E]\!]^M(S, \ldots, S)$ *for some* $M \in \mathbb{N}$

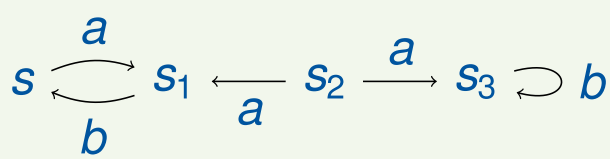
Proof.

omitted □

Mutually Recursive Equational Systems

A Mutually Recursive Specification

Example 6.6



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

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

Computation of $\mathrm{FIX}(\llbracket E \rrbracket)$: on the board

Outline of Lecture 6

Recap: Fixed-Point Theory

Largest Fixed Points and Invariants

Mutually Recursive Equational Systems

Mixing Least and Greatest Fixed Points

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 6.7

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

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Mixing Least and Greatest Fixed Points I

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

Example 6.7

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

can be specified by

$$Pos(Livelock)$$

where

$$Pos(F) \stackrel{min}{=} F \vee \langle Act \rangle Pos(F) \qquad \text{(cf. Example 4.4)}$$
$$Livelock \stackrel{max}{=} \langle \tau \rangle Livelock$$

(thus, $Livelock \equiv Forever(\tau)$ [cf. Example 6.2])

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 6.8

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

Mixing Least and Greatest Fixed Points II

Caveat: arbitrary mixing can entail non-monotonic behaviour

Example 6.8

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

Fixed-point iteration:

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

Mixing Least and Greatest Fixed Points II

Caveat: arbitrary mixing can entail non-monotonic behaviour

Example 6.8

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

Fixed-point iteration:

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

Mixing Least and Greatest Fixed Points II

Caveat: arbitrary mixing can entail non-monotonic behaviour

Example 6.8

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

Fixed-point iteration:

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

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Mixing Least and Greatest Fixed Points II

Caveat: arbitrary mixing can entail non-monotonic behaviour

Example 6.8

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

Fixed-point iteration:

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

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 6.8

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

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

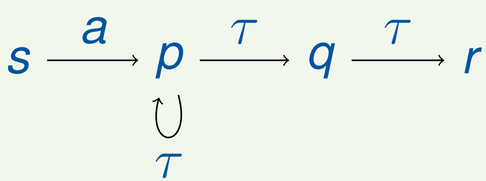
- are of same type (either *min* or *max*) and
- use only variables defined in the same or subsequent blocks

$\Longrightarrow$ bottom-up, block-wise evaluation by fixed-point iteration

Mixing Least and Greatest Fixed Points III

Example 6.9 (cf. Example 6.7)

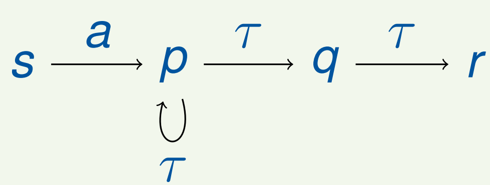
$$PosLL \stackrel{min}{=} Livelock \vee \langle Act \rangle PosLL$$
$$Livelock \stackrel{max}{=} \langle \tau \rangle Livelock$$



Mixing Least and Greatest Fixed Points III

Example 6.9 (cf. Example 6.7)

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



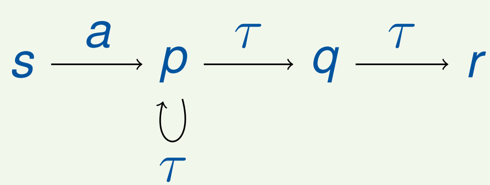
1. Fixed-point iteration for $Livelock : T \mapsto \langle \cdot\tau\cdot \rangle(T)$:

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

Mixing Least and Greatest Fixed Points III

Example 6.9 (cf. Example 6.7)

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



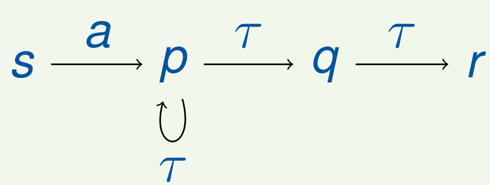
1. Fixed-point iteration for $\mathit{Livelock} : T \mapsto \langle \cdot \tau \cdot \rangle (T)$:

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

Mixing Least and Greatest Fixed Points III

Example 6.9 (cf. Example 6.7)

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



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

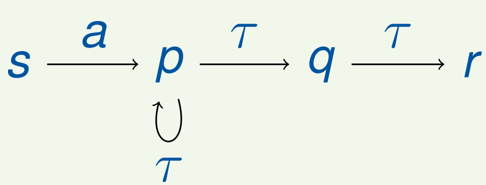
$$S = \{s, p, q, r\} \mapsto \{p, q\} \mapsto \{p\}$$

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Mixing Least and Greatest Fixed Points III

Example 6.9 (cf. Example 6.7)

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



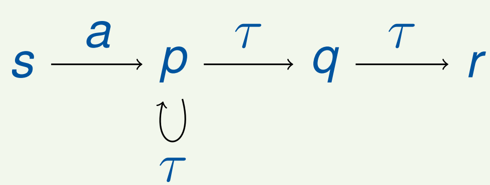
1. Fixed-point iteration for $Livelock : T \mapsto \langle \cdot \tau \cdot \rangle (T)$:

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

Mixing Least and Greatest Fixed Points III

Example 6.9 (cf. Example 6.7)

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



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

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

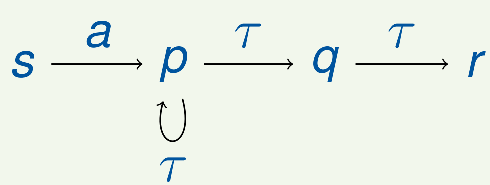
2. Fixed-point iteration for $PosLL : T \mapsto \{p\} \cup \langle \cdot Act \cdot \rangle (T)$:

$$\emptyset$$

Mixing Least and Greatest Fixed Points III

Example 6.9 (cf. Example 6.7)

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



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

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

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

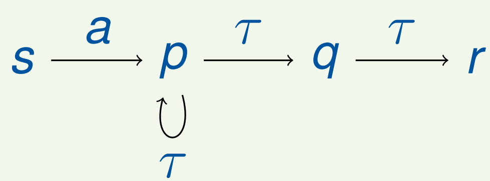
$$\emptyset \mapsto \{p\}$$

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Mixing Least and Greatest Fixed Points III

Example 6.9 (cf. Example 6.7)

$$\begin{aligned} PosLL &\overset{min}{=} Livelock \vee \langle Act \rangle PosLL \\ Livelock &\overset{max}{=} \langle \tau \rangle Livelock \end{aligned}$$



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

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

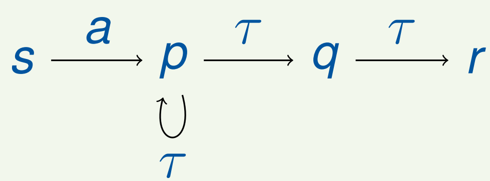
2. Fixed-point iteration for $PosLL : T \mapsto \{p\} \cup \langle \cdot Act \cdot \rangle (T)$:

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

Mixing Least and Greatest Fixed Points III

Example 6.9 (cf. Example 6.7)

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



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

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

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

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

Mixing Least and Greatest Fixed Points

The Modal μ-Calculus

- Logic that supports free mixing of least and greatest fixed points:
  - D. Kozen: *Results on the Propositional μ-Calculus*, Theoretical Computer Science 27, 1983, 333–354

Mixing Least and Greatest Fixed Points

The Modal μ-Calculus

- Logic that supports free mixing of least and greatest fixed points:
  - D. Kozen: *Results on the Propositional μ-Calculus*, Theoretical Computer Science 27, 1983, 333–354
- 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, 133–153

The Modal μ-Calculus

- Logic that supports free mixing of least and greatest fixed points:
  - D. Kozen: *Results on the Propositional μ-Calculus*, Theoretical Computer Science 27, 1983, 333–354
- 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, 133–153
- 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, ...)

Mixing Least and Greatest Fixed Points

The Modal μ-Calculus

- Logic that supports free mixing of least and greatest fixed points:
  - D. Kozen: *Results on the Propositional μ-Calculus*, Theoretical Computer Science 27, 1983, 333–354
- 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, 133–153
- 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, 545–623