Theoretical Foundations of the UML

Lecture 15+16: A Logic for MSCs

Joost-Pieter Katoen

Lehrstuhl für Informatik 2
Software Modeling and Verification Group

moves.rwth-aachen.de/teaching/ss-16/theoretical-foundations-of-the-uml/

26. Juni 2016

RWTH AACHEN UNIVERSITY

Outline

Overview

1. Introduction
2. Local Formulas and Path Expressions
   - Syntax
   - Formal Semantics
3. PDL Formulas
4. Verification problems for PDL
   - Model checking MSCs
   - Model checking CFMs
   - Model checking MSGs
   - Satisfiability

A logic for MSCs

- This lecture will be devoted to a logic that is interpreted over MSCs

A logic for MSCs

- This lecture will be devoted to a logic that is interpreted over MSCs

A logic for MSCs

- This lecture will be devoted to a logic that is interpreted over MSCs
- The logic is used to umambigously express properties of MSCs
  - does a given MSC M satisfy the logical formula φ?

A logic for MSCs

- This lecture will be devoted to a logic that is interpreted over MSCs
- The logic is used to umambigously express properties of MSCs
  - does a given MSC M satisfy the logical formula φ?
- And to characterise a set of MSCs by means of a logical formula
  - all MSCs that satisfy the formula φ

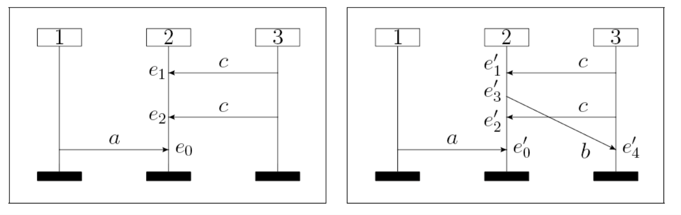
A logic for MSCs

- This lecture will be devoted to a logic that is interpreted over MSCs
- The logic is used to umambigously express properties of MSCs
  - does a given MSC M satisfy the logical formula φ?
- And to characterise a set of MSCs by means of a logical formula
  - all MSCs that satisfy the formula φ
- Based on propositional dynamic logic (PDL) [Fischer & Ladner, 1979]
  - combines easy-to-grasp concepts such as regular expressions and Boolean operators

A logic for MSCs

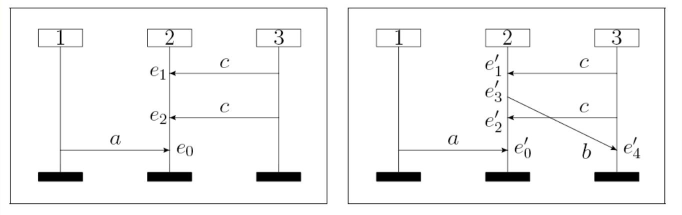
- This lecture will be devoted to a logic that is interpreted over MSCs
- The logic is used to umambigously express properties of MSCs
  - does a given MSC M satisfy the logical formula φ?
- And to characterise a set of MSCs by means of a logical formula
  - all MSCs that satisfy the formula φ
- Based on propositional dynamic logic (PDL) [Fischer & Ladner, 1979]
  - combines easy-to-grasp concepts such as regular expressions and Boolean operators
- Syntax, semantics, examples and various verification problems.

Some informal example properties



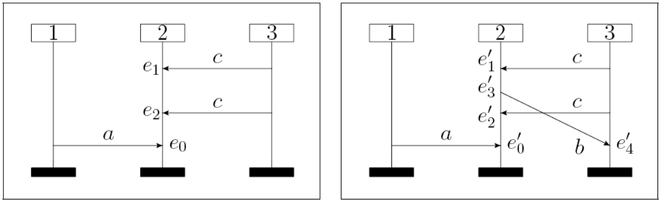
1. The (unique) maximal event of M is labeled by $?(2, 1, a)$ Yes. No.

Some informal example properties



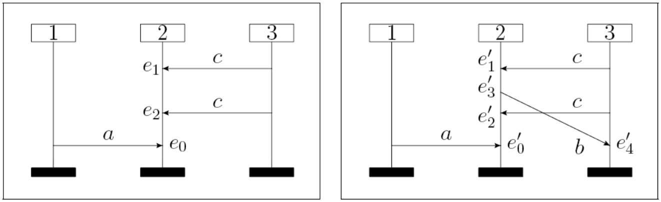
1. The (unique) maximal event of M is labeled by $?(2,1,a)$ Yes. No.
2. The maximal event on process 2 is labeled by $?(2,1,a)$ Yes. Yes.

Some informal example properties



1. The (unique) maximal event of M is labeled by $?(2, 1, a)$ Yes. No.
2. The maximal event on process 2 is labeled by $?(2, 1, a)$ Yes. Yes.
3. No two consecutive events are labeled with $?(2, 3, c)$ No. Yes.

Some informal example properties



1. The (unique) maximal event of M is labeled by $?(2, 1, a)$ Yes. No.
2. The maximal event on process 2 is labeled by $?(2, 1, a)$ Yes. Yes.
3. No two consecutive events are labeled with $?(2, 3, c)$ No. Yes.
4. The number of send events at process 3 is odd. No. No.

The need for logics

The need for logics

- Properties stated in natural language are ambiguous.

The need for logics

- Properties stated in natural language are ambiguous.

- We prefer to use a formal language for expressing properties.

The need for logics

- Properties stated in natural language are ambiguous.
- We prefer to use a formal language for expressing properties.
- A formal semantics yields an unambiguous interpretation.

The need for logics

- Properties stated in natural language are ambiguous.
- We prefer to use a formal language for expressing properties.
- A formal semantics yields an unambiguous interpretation.
- This provides the basis for verification algorithms and common understanding.

The need for logics

- Properties stated in natural language are ambiguous.
- We prefer to use a formal language for expressing properties.
- A formal semantics yields an unambiguous interpretation.
- This provides the basis for verification algorithms and common understanding.
- As formal language for properties we use logic.

Overview

The logic PDL

The logic PDL

- Local formulas
  - Statements interpreted for single events in an MSC
  - Express properties about other events at the same process
  - Express properties about send and matched receive events

The logic PDL

- Local formulas
  - Statements interpreted for single events in an MSC
  - Express properties about other events at the same process
  - Express properties about send and matched receive events
- Path expressions
  - Used to navigate through an MSC
  - Use choice, concatenation and repetition
  - Can be embraced in box and diamond modalities

The logic PDL

- Local formulas
  - Statements interpreted for single events in an MSC
  - Express properties about other events at the same process
  - Express properties about send and matched receive events
- Path expressions
  - Used to navigate through an MSC
  - Use choice, concatenation and repetition
  - Can be embraced in box and diamond modalities
- PDL-formulas
  - Express properties about an entire MSC

Local formulas

Local formulas

Local formulas

These are statements over single events in an MSC. That is, an event either satisfies or refutes such formula.

Local formulas

Local formulas

These are statements over single events in an MSC. That is, an event either satisfies or refutes such formula.

Example local formulas

Local formulas

Local formulas

These are statements over single events in an MSC. That is, an event either satisfies or refutes such formula.

Example local formulas

- $!(1, 2, a)$ — The current event is labeled with $!(1, 2, a)$

Local formulas

Local formulas

These are statements over single events in an MSC. That is, an event either satisfies or refutes such formula.

Example local formulas

- $!(1, 2, a)$ — The current event is labeled with $!(1, 2, a)$
- $\langle \mathsf{proc} \rangle \mathit{true}$ — There is a next event at the same process

Local formulas

Local formulas

These are statements over single events in an MSC. That is, an event either satisfies or refutes such formula.

Example local formulas

- $!(1, 2, a)$ — The current event is labeled with $!(1, 2, a)$
- $\langle \mathsf{proc} \rangle true$ — There is a next event at the same process
- $\langle \mathsf{proc}; \mathsf{proc} \rangle true$ — There are (at least) two next events at this process

Local formulas

Local formulas

These are statements over single events in an MSC. That is, an event either satisfies or refutes such formula.

Example local formulas

- $!(1, 2, a)$ — The current event is labeled with $!(1, 2, a)$
- $\langle \mathsf{proc} \rangle \mathit{true}$ — There is a next event at the same process
- $\langle \mathsf{proc}; \mathsf{proc} \rangle \mathit{true}$ — There are (at least) two next events at this process
- $[\mathsf{proc}]^{-1} \mathit{false}$ — There is no preceding event at this process

Local formulas

Local formulas

These are statements over single events in an MSC. That is, an event either satisfies or refutes such formula.

Example local formulas

- $!(1, 2, a)$ — The current event is labeled with $!(1, 2, a)$
- $\langle \mathsf{proc} \rangle \mathit{true}$ — There is a next event at the same process
- $\langle \mathsf{proc}; \mathsf{proc} \rangle \mathit{true}$ — There are (at least) two next events at this process
- $[\mathsf{proc}]^{-1} \mathit{false}$ — There is no preceding event at this process
- $\langle \mathsf{msg} \rangle \mathit{true}$ — This event is a send matching a (next) receive event

Local formulas

Local formulas

These are statements over single events in an MSC. That is, an event either satisfies or refutes such formula.

Example local formulas

- $!(1, 2, a)$ — The current event is labeled with $!(1, 2, a)$
- $\langle \textsf{proc} \rangle \mathit{true}$ — There is a next event at the same process
- $\langle \textsf{proc}; \textsf{proc} \rangle \mathit{true}$ — There are (at least) two next events at this process
- $[\textsf{proc}]^{-1} \mathit{false}$ — There is no preceding event at this process
- $\langle \textsf{msg} \rangle \mathit{true}$ — This event is a send matching a (next) receive event
- $\langle \textsf{proc} \rangle \, ?(1, 2, b)$ — Event $?(1, 2, b)$ is a possible next event on this process

Local formulas

Local formulas

These are statements over single events in an MSC. That is, an event either satisfies or refutes such formula.

Example local formulas

- $!(1,2,a)$ — The current event is labeled with $!(1,2,a)$
- $\langle \textsf{proc} \rangle \mathit{true}$ — There is a next event at the same process
- $\langle \textsf{proc}; \textsf{proc} \rangle \mathit{true}$ — There are (at least) two next events at this process
- $[\textsf{proc}]^{-1} \mathit{false}$ — There is no preceding event at this process
- $\langle \textsf{msg} \rangle \mathit{true}$ — This event is a send matching a (next) receive event
- $\langle \textsf{proc} \rangle\, ?(1,2,b)$ — Event $?(1,2,b)$ is a possible next event on this process
- $[\{\, \neg !(1,2,a)\, \}] \mathit{true}$ — An event is possible after any event different from $!(1,2,a)$

Local formulas

Definition (Syntax of local formulas)

For communication action $\sigma \in Act$ and path expression α, the grammar of local formulas is given by:

$$\varphi \; ::= \; true \; \mid \; \sigma \; \mid \; \neg\varphi \; \mid \; \varphi \vee \varphi \; \mid \; \langle\alpha\rangle\varphi \; \mid \; \langle\alpha\rangle^{-1}\varphi$$

The syntax of path expressions α will be defined later on.

Local formulas

Definition (Syntax of local formulas)

For communication action $\sigma \in Act$ and path expression α, the grammar of local formulas is given by:

$$\varphi \ ::= \ true \ | \ \sigma \ | \ \neg\varphi \ | \ \varphi \vee \varphi \ | \ \langle\alpha\rangle\varphi \ | \ \langle\alpha\rangle^{-1}\varphi$$

The syntax of path expressions α will be defined later on.

Definition (Derived operators)

Local formulas

Definition (Syntax of local formulas)

For communication action $\sigma \in Act$ and path expression α, the grammar of local formulas is given by:

$$\varphi \ ::= \ true \ | \ \sigma \ | \ \neg\varphi \ | \ \varphi \vee \varphi \ | \ \langle\alpha\rangle\varphi \ | \ \langle\alpha\rangle^{-1}\varphi$$

The syntax of path expressions α will be defined later on.

Definition (Derived operators)

$$\begin{aligned}
false &:= \neg true \\
\varphi_1 \wedge \varphi_2 &:= \neg(\neg\varphi_1 \vee \neg\varphi_2) \\
\varphi_1 \rightarrow \varphi_2 &:= \neg\varphi_1 \vee \varphi_2 \\
[\alpha]\varphi &:= \neg\langle\alpha\rangle\neg\varphi \\
[\alpha]^{-1}\varphi &:= \neg\langle\alpha\rangle^{-1}\neg\varphi
\end{aligned}$$

Intuitive meaning of local formulas

Intuitive meaning of local formulas

$true$ Valid statement. Satisfied by every event.

Intuitive meaning of local formulas

$true$	Valid statement. Satisfied by every event.
σ	Current event is labelled with σ

Intuitive meaning of local formulas

$true$	Valid statement. Satisfied by every event.
σ	Current event is labelled with σ
$\neg\varphi$	Current event does not satisfy φ

Intuitive meaning of local formulas

$true$	Valid statement. Satisfied by every event.
σ	Current event is labelled with σ
$\neg\varphi$	Current event does not satisfy φ
$\varphi_1 \vee \varphi_2$	Current event satisfies φ_1 or φ_2

Intuitive meaning of local formulas

$true$	Valid statement. Satisfied by every event.
σ	Current event is labelled with σ
$\neg\varphi$	Current event does not satisfy φ
$\varphi_1 \vee \varphi_2$	Current event satisfies φ_1 or φ_2
$\langle\alpha\rangle\varphi$	Some forward path satisfying α reaches an event satisfying φ

Intuitive meaning of local formulas

true	Valid statement. Satisfied by every event.
σ	Current event is labelled with σ
$\neg\varphi$	Current event does not satisfy φ
$\varphi_1 \vee \varphi_2$	Current event satisfies φ_1 or φ_2
$\langle\alpha\rangle\varphi$	Some forward path satisfying α reaches an event satisfying φ
$\langle\alpha\rangle^{-1}\varphi$	Some backward path α reaches an event satisfying φ

Intuitive meaning of local formulas

$true$	Valid statement. Satisfied by every event.
σ	Current event is labelled with σ
$\neg\varphi$	Current event does not satisfy φ
$\varphi_1 \vee \varphi_2$	Current event satisfies φ_1 or φ_2
$\langle\alpha\rangle\varphi$	Some forward path satisfying α reaches an event satisfying φ
$\langle\alpha\rangle^{-1}\varphi$	Some backward path α reaches an event satisfying φ
$[\alpha]\varphi$	All forward paths satisfying α reach an event satisfying φ

Intuitive meaning of local formulas

true	Valid statement. Satisfied by every event.
σ	Current event is labelled with σ
$\neg\varphi$	Current event does not satisfy φ
$\varphi_1 \vee \varphi_2$	Current event satisfies φ_1 or φ_2
$\langle\alpha\rangle\varphi$	Some forward path satisfying α reaches an event satisfying φ
$\langle\alpha\rangle^{-1}\varphi$	Some backward path α reaches an event satisfying φ
$[\alpha]\varphi$	All forward paths satisfying α reach an event satisfying φ
$[\alpha]^{-1}\varphi$	All backward paths satisfying α reach an event satisfying φ

Intuitive meaning of local formulas

true	Valid statement. Satisfied by every event.
σ	Current event is labelled with σ
$\neg\varphi$	Current event does not satisfy φ
$\varphi_1 \vee \varphi_2$	Current event satisfies φ_1 or φ_2
$\langle\alpha\rangle\varphi$	Some forward path satisfying α reaches an event satisfying φ
$\langle\alpha\rangle^{-1}\varphi$	Some backward path α reaches an event satisfying φ
$[\alpha]\varphi$	All forward paths satisfying α reach an event satisfying φ
$[\alpha]^{-1}\varphi$	All backward paths satisfying α reach an event satisfying φ

How are path expressions like α defined?

Path expressions

Definition (Syntax of local formulas)

For communication action $\sigma \in Act$ and path expression α, the grammar of local formulas is given by:

$$\varphi \ ::= \ true \ \mid \ \sigma \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \langle\alpha\rangle\varphi \ \mid \ \langle\alpha\rangle^{-1}\varphi$$

Path expressions

Definition (Syntax of local formulas)

For communication action $\sigma \in Act$ and path expression α, the grammar of local formulas is given by:

$$\varphi \ ::= \ true \ | \ \sigma \ | \ \neg\varphi \ | \ \varphi \vee \varphi \ | \ \langle\alpha\rangle\varphi \ | \ \langle\alpha\rangle^{-1}\varphi$$

Definition (Syntax of path expressions)

For local formula φ, the grammar of path expressions is given by:

$$\alpha \ ::= \ \{\varphi\} \ | \ \mathsf{proc} \ | \ \mathsf{msg} \ | \ \alpha;\alpha \ | \ \alpha+\alpha \ | \ \alpha^{*}$$

Intuitive meaning of path expressions

- $\{\varphi\}$ specifies an event that satisfies φ

Intuitive meaning of path expressions

- $\{ \varphi \}$ specifies an event that satisfies φ
- proc requires a (direct) successor relation between events at the same process

Intuitive meaning of path expressions

- $\{\varphi\}$ specifies an event that satisfies φ
- proc requires a (direct) successor relation between events at the same process
- msg requires a matching between current event and a receive event

Intuitive meaning of path expressions

- $\{ \varphi \}$ specifies an event that satisfies φ
- proc requires a (direct) successor relation between events at the same process
- msg requires a matching between current event and a receive event
- The composition $\alpha; \beta$ defines the set of pairs (e, e') for which there exist event e'' such that $(e, e'') \models \alpha$ and $(e'', e') \models \beta$

Intuitive meaning of path expressions

- $\{ \varphi \}$ specifies an event that satisfies φ
- proc requires a (direct) successor relation between events at the same process
- msg requires a matching between current event and a receive event
- The composition $\alpha; \beta$ defines the set of pairs (e, e') for which there exist event e'' such that $(e, e'') \models \alpha$ and $(e'', e') \models \beta$
- $\alpha + \beta$ denotes the union of the relations α and β

Intuitive meaning of path expressions

- $\{ \varphi \}$ specifies an event that satisfies φ
- proc requires a (direct) successor relation between events at the same process
- msg requires a matching between current event and a receive event
- The composition $\alpha ; \beta$ defines the set of pairs (e, e') for which there exist event e'' such that $(e, e'') \models \alpha$ and $(e'', e') \models \beta$
- $\alpha + \beta$ denotes the union of the relations α and β
- α^* denotes the reflexive and transitive closure of the relation α

Intuitive meaning of local formulas

- Local formulas are interpreted over MSC events

Intuitive meaning of local formulas

- Local formulas are interpreted over MSC events
- Event e satisfies $\underbrace{!(p, q, a)}_{\sigma}$ iff e is labelled with action $\underbrace{!(p, q, a)}_{\sigma}$

Intuitive meaning of local formulas

- Local formulas are interpreted over MSC events
- Event e satisfies $\underbrace{!(p,q,a)}_{\sigma}$ iff e is labelled with action $\underbrace{!(p,q,a)}_{\sigma}$
- Path expression α defines a binary relation between events:

Intuitive meaning of local formulas

- Local formulas are interpreted over MSC events
- Event e satisfies $\underbrace{!(p, q, a)}_{\sigma}$ iff e is labelled with action $\underbrace{!(p, q, a)}_{\sigma}$
- Path expression α defines a binary relation between events:
  1. $\{\varphi\}$ is the set of pairs (e, e') such that e satisfies φ

Intuitive meaning of local formulas

- Local formulas are interpreted over MSC events
- Event e satisfies $\underbrace{!(p, q, a)}_{\sigma}$ iff e is labelled with action $\underbrace{!(p, q, a)}_{\sigma}$
- Path expression α defines a binary relation between events:
  1. $\{\varphi\}$ is the set of pairs (e, e') such that e satisfies φ
  2. $(e, e') \models \mathsf{proc}$ iff e and e' reside at the same process (p, say) and e' is a direct successor of e wrt. $<_p$

Intuitive meaning of local formulas

- Local formulas are interpreted over MSC events
- Event e satisfies $\underbrace{!(p, q, a)}_{\sigma}$ iff e is labelled with action $\underbrace{!(p, q, a)}_{\sigma}$
- Path expression α defines a binary relation between events:
  1. $\{\varphi\}$ is the set of pairs (e, e') such that e satisfies φ
  2. $(e, e') \models \mathsf{proc}$ iff e and e' reside at the same process (p, say) and e' is a direct successor of e wrt. $<_p$
  3. $(e, e') \models \mathsf{msg}$ iff e' is the matching event of e, i.e., $e' = m(e)$

Forward and backward local formulas

Forward and backward local formulas

- Event e satisfies $\langle \alpha \rangle \varphi$ iff there is an event e' such that (e, e') satisfies α and e' satisfies φ

Forward and backward local formulas

- Event e satisfies $\langle\alpha\rangle\varphi$ iff there is an event e' such that (e, e') satisfies α and e' satisfies φ

  Formula $\langle\alpha\rangle\varphi$ looks "forward" along the partial order of the MSC starting from the current event

Forward and backward local formulas

- Event e satisfies $\langle \alpha \rangle \varphi$ iff there is an event e' such that (e, e') satisfies α and e' satisfies φ

  Formula $\langle \alpha \rangle \varphi$ looks "forward" along the partial order of the MSC starting from the current event

- The interpretation of $\langle \alpha \rangle^{-1} \varphi$ is dual, i.e., e satisfies it iff there is an event e' such that (e', e) satisfies α and e' satisfies φ

Forward and backward local formulas

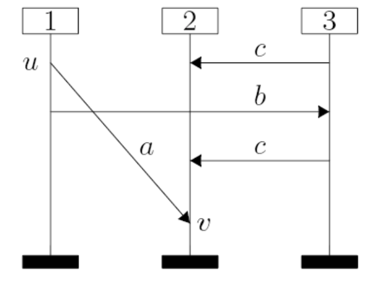
- Event e satisfies $\langle \alpha \rangle \varphi$ iff there is an event e' such that (e, e') satisfies α and e' satisfies φ

  Formula $\langle \alpha \rangle \varphi$ looks "forward" along the partial order of the MSC starting from the current event

- The interpretation of $\langle \alpha \rangle^{-1} \varphi$ is dual, i.e., e satisfies it iff there is an event e' such that (e', e) satisfies α and e' satisfies φ

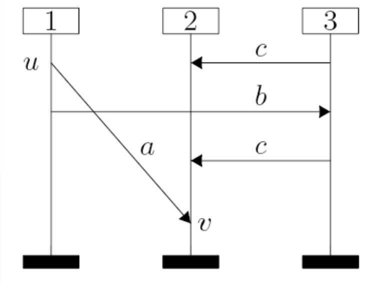
  Formula $\langle \alpha \rangle^{-1} \varphi$ looks "backward" along the partial order of the MSC starting from the current event

Example



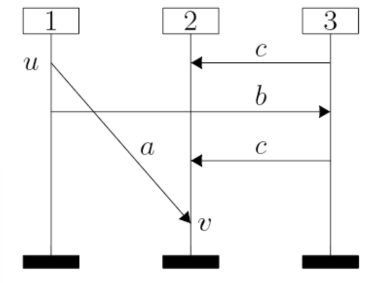
1. $u \models !(1, 2, a)$ u is labelled with the action $!(1, 2, a)$

Example



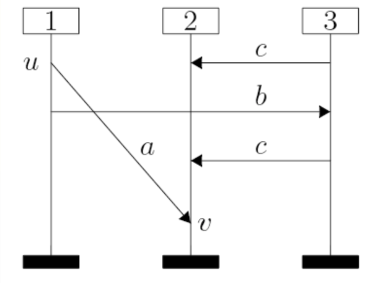
1. $u \models\, !(1,2,a)$ — u is labelled with the action $!(1,2,a)$
2. $u \models [\mathsf{proc}]^{-1}\, \mathit{false}$ — u is the first event on u's process

Example



1. $u \models\, !(1,2,a)$ — u is labelled with the action $!(1,2,a)$
2. $u \models [\mathsf{proc}]^{-1} \mathit{false}$ — u is the first event on u's process
3. $u \models \langle \mathsf{msg} \rangle ?(2,1,a)$ — event u matches with the event v

Example



1. $u \models !(1,2,a)$ — u is labelled with the action $!(1,2,a)$
2. $u \models [\mathsf{proc}]^{-1}\, \mathit{false}$ — u is the first event on u's process
3. $u \models \langle \mathsf{msg} \rangle ?(2,1,a)$ — event u matches with the event v
4. $u \models \langle (\mathsf{proc} + \mathsf{msg})^* \rangle !(3,2,c)$ — event u happens before $!(3,2,c)$

Semantics of local formulas (1)

Definition (Syntax of local formulas)

For communication action $\sigma \in Act$ and path expression α:

$$\varphi \ ::= \ true \ \mid \ \sigma \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \langle\alpha\rangle\varphi \ \mid \ \langle\alpha\rangle^{-1}\varphi$$

Semantics of local formulas (1)

Definition (Syntax of local formulas)

For communication action $\sigma \in Act$ and path expression α:

$$\varphi \ ::= \ true \ \mid \ \sigma \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \langle\alpha\rangle\varphi \ \mid \ \langle\alpha\rangle^{-1}\varphi$$

Definition (Semantics of base local formulas)

Let $M = (\mathcal{P}, E, \mathcal{C}, l, m, <) \in \mathbb{M}$ be an MSC and $e \in E$.

Binary relation $\models$ is defined such that $((M, e), \varphi) \in \models$ iff event e of MSC M satisfies local formula φ. We write $M, e \models \varphi$ for $((M, e), \varphi) \in \models$.

$$\begin{array}{rcl} M, e \models true & & \text{for all } e \in E \\ M, e \models \sigma & \text{iff} & l(e) = \sigma \\ M, e \models \neg\varphi & \text{iff} & \text{not } M, e \models \varphi \\ M, e \models \varphi_1 \vee \varphi_2 & \text{iff} & M, e \models \varphi_1 \text{ or } M, e \models \varphi_2 \end{array}$$

Semantics of local formulas (2)

Definition (Semantics of forward path formulas)

Let $M = (\mathcal{P}, E, \mathcal{C}, l, m, <) \in \mathbb{M}$ be an MSC and $e \in E$.

$$
\begin{array}{rcl}
e \models \langle\{\psi\}\rangle\varphi & \text{iff} & e \models \psi \text{ and } e \models \varphi \\
e \models \langle\mathsf{proc}\rangle\varphi & \text{iff} & \exists e' \in E.\, e \lessdot_p e' \text{ and } e' \models \varphi \\
e \models \langle\mathsf{msg}\rangle\varphi & \text{iff} & \exists e' \in E.\, e' = m(e) \text{ and } e' \models \varphi \\
e \models \langle\alpha_1;\alpha_2\rangle\varphi & \text{iff} & e \models \langle\alpha_1\rangle\langle\alpha_2\rangle\varphi \\
e \models \langle\alpha_1 + \alpha_2\rangle\varphi & \text{iff} & e \models \langle\alpha_1\rangle\varphi \text{ or } e \models \langle\alpha_2\rangle\varphi \\
e \models \langle\alpha^*\rangle\varphi & \text{iff} & \exists n \in \mathbb{N}.\, e \models (\langle\alpha\rangle)^n\,\varphi
\end{array}
$$

Where $e \lessdot_p e'$ iff $e <_p e'$ and $\neg(\exists e''.\, e <_p e'' <_p e')$, i.e., e' is a direct successor of e under $<_p$.

Semantics of local formulas (3)

Definition (Semantics of backward path formulas)

Let $M = (\mathcal{P}, E, \mathcal{C}, l, m, <) \in \mathbb{M}$ be an MSC and $e \in E$.

$$
\begin{array}{rcl}
e \models \langle \{\psi\} \rangle^{-1} \varphi & \text{iff} & e \models \psi \text{ and } e \models \varphi \\
e \models \langle \mathsf{proc} \rangle^{-1} \varphi & \text{iff} & \exists e' \in E.\, e' <\!\!\cdot_p\, e \text{ and } e' \models \varphi \\
e \models \langle \mathsf{msg} \rangle^{-1} \varphi & \text{iff} & \exists e' \in E.\, e' = m^{-1}(e) \text{ and } e' \models \varphi \\
e \models \langle \alpha_1 ; \alpha_2 \rangle^{-1} \varphi & \text{iff} & e \models \langle \alpha_1 \rangle^{-1} \langle \alpha_2 \rangle^{-1} \varphi \\
e \models \langle \alpha_1 + \alpha_2 \rangle^{-1} \varphi & \text{iff} & e \models \langle \alpha_1 \rangle^{-1} \varphi \text{ or } e \models \langle \alpha_2 \rangle^{-1} \varphi \\
e \models \langle \alpha^* \rangle^{-1} \varphi & \text{iff} & \exists n \in \mathbb{N}.\, e \models \left(\langle \alpha \rangle^{-1} \right)^n \varphi
\end{array}
$$

Overview

PDL formulas

Definition (Syntax of PDL formulas)

For local formula φ, the grammar of PDL formulas is given by:

$$\Phi \ ::= \ \exists \varphi \ \mid \ \forall \varphi \ \mid \ \Phi \wedge \Phi \ \mid \ \Phi \vee \Phi$$

Negation

Negation is absent. As existential and universal quantification, as well as conjunction and disjunction are present, PDF-formulas are closed under negation.

Intuitive meaning of PDL formulas

- MSC M satisfies $\exists\varphi$ if M has some event e satisfying φ

Intuitive meaning of PDL formulas

- MSC M satisfies $\exists\varphi$ if M has some event e satisfying φ
- MSC M satisfies $\exists\langle\alpha\rangle\varphi$ if from some event e in M, there exists an α-labelled path from e to an event e', say, satisfying φ

Intuitive meaning of PDL formulas

- MSC M satisfies $\exists\varphi$ if M has some event e satisfying φ
- MSC M satisfies $\exists\langle\alpha\rangle\varphi$ if from some event e in M, there exists an α-labelled path from e to an event e', say, satisfying φ
- MSC M satisfies $\exists[\alpha]\varphi$ if from some event e in M, every event that can be reached via an α-labelled path satisfies φ

Semantics of PDL formulas

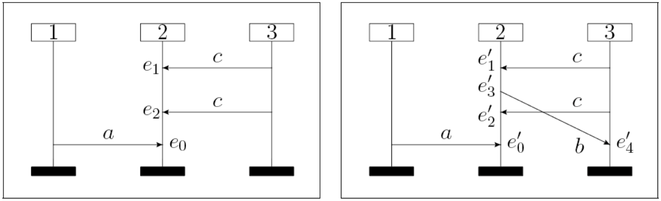
Definition (Semantics of PDL formulas)

Let $M = (\mathcal{P}, E, \mathcal{C}, l, m, <) \in \mathbb{M}$ be an MSC.

$(M, \Phi) \in \models$ iff PDL formula Φ holds in MSC M.

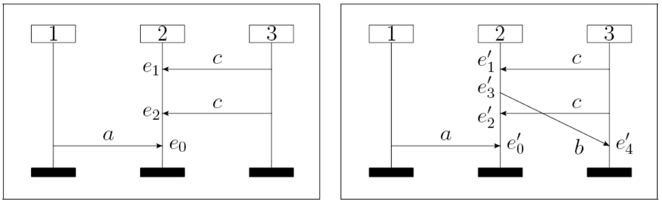
$$
\begin{aligned}
M &\models \exists\varphi && \text{iff} && \exists e \in E.\, M, e \models \varphi \\
M &\models \forall\varphi && \text{iff} && \forall e \in E.\, M, e \models \varphi \\
M &\models \Phi_1 \wedge \Phi_2 && \text{iff} && M \models \Phi_1 \text{ and } M \models \Phi_2 \\
M &\models \Phi_1 \vee \Phi_2 && \text{iff} && M \models \Phi_1 \text{ or } M \models \Phi_2
\end{aligned}
$$

Example (1)

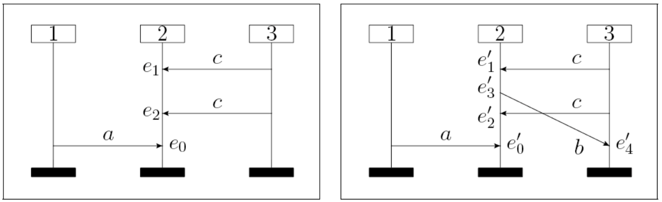


- The (unique) maximal event of M is labeled by $?(2, 1, a)$ Yes. No.

Example (1)

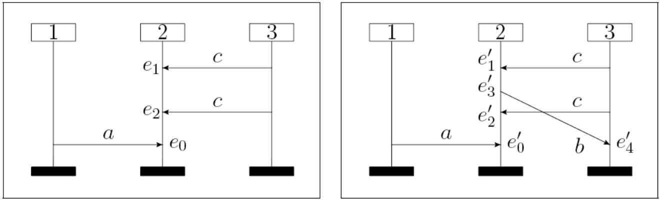


- The (unique) maximal event of M is labeled by $?(2, 1, a)$ Yes. No.
- $\forall\,(\langle(\mathsf{proc} + \mathsf{msg})^*\rangle([\mathsf{proc}]\,\mathit{false} \wedge ?(2, 1, a)))$ Yes. No.



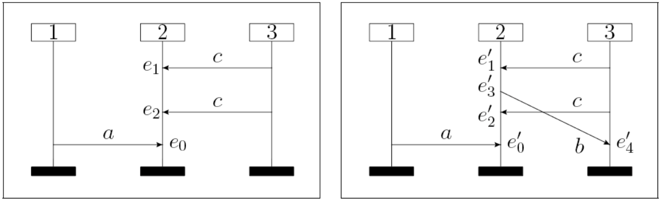
- The maximal event on process 2 is labeled by $?(2, 1, a)$ Yes. Yes.

Example (2)



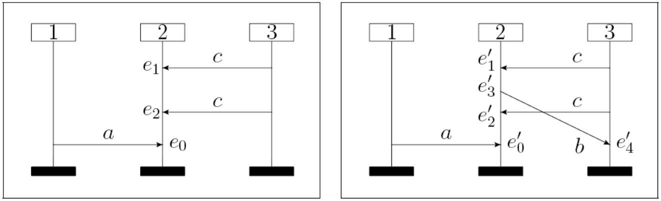
- The maximal event on process 2 is labeled by $?(2,1,a)$ Yes. Yes.
- $\exists\,([\mathsf{proc}]\, \mathit{false} \wedge\, ?(2,1,a))$ Yes. Yes.

Example (3)



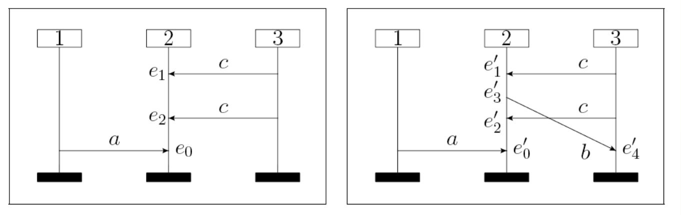
- No two consecutive events are labeled with $?(2, 3, c)$ No. Yes.

Example (3)



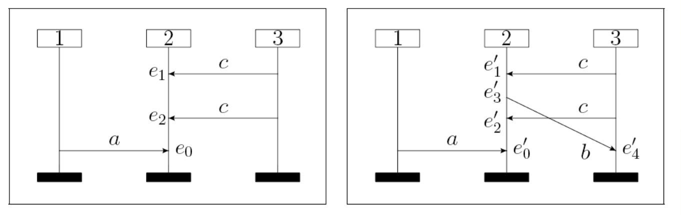
- No two consecutive events are labeled with $?(2, 3, c)$ No. Yes.
- $\forall\,([\{\,?(2,3,c)\,\}; \mathsf{proc}; \{\,?(2,3,c)\,\}]\,\mathit{false})$ No. Yes.

Example (4)



- The number of send events at process 3 is odd. No. No.

Example (4)



- The number of send events at process 3 is odd. No. No.

- See next slide

Example

MSC M has an even number of messages sent from process 1 to 2:

$$\forall\Big(\underbrace{[\mathsf{proc}]^{-1} \mathit{false} \wedge P_1}_{\text{minimal event on process 1}} \rightarrow \langle\alpha\rangle \underbrace{[\mathsf{proc}]\, \mathit{false}}_{\text{maximal event on process}} \Big)$$

where $P_1 = \bigvee_{j \in \mathcal{P}, j \neq 1} (!_{1,j} \vee ?_{1,j})$ with $!_{1,j} = \bigvee_{a \in \mathcal{C}} !(1, j, a)$ and $?_{1,j}$ is defined in a similar way, i.e., $e \models P_1$ iff e occurs at process 1.

Path expression α is defined by:

$$\alpha \;=\; ((\{\neg !_1\}; \mathsf{proc})^*; \{!_1\}; \mathsf{proc}; (\{\neg !_1\}; \mathsf{proc})^*; \{!_1\}; \mathsf{proc}; (\{\neg !_1\}; \mathsf{proc})^*)^*$$

and where $!_1$ abbreviates $\bigvee_{a \in \mathcal{C}} !(1, 2, a)$

Overview

Model checking MSCs

Model checking MSCs versus PDL [Kern, 2009]

Model checking MSCs

Model checking MSCs versus PDL [Kern, 2009]

The following model-checking problem is decidable in polynomial time:

INPUT: MSC M, PDL-formula Φ

OUTPUT: does $M \models \Phi$?

Model checking MSCs

Model checking MSCs versus PDL [Kern, 2009]

The following model-checking problem is decidable in polynomial time:

INPUT: MSC M, PDL-formula Φ

OUTPUT: does $M \models \Phi$?

Model checking MSCs

Model checking MSCs versus PDL [Kern, 2009]

The following model-checking problem is decidable in polynomial time:

INPUT: MSC M, PDL-formula Φ

OUTPUT: does $M \models \Phi$?

Proof.

(Sketch). Let Φ be a PDL formula.

Model checking MSCs

Model checking MSCs versus PDL [Kern, 2009]

The following model-checking problem is decidable in polynomial time:

INPUT: MSC M, PDL-formula Φ

OUTPUT: does $M \models \Phi$?

Proof.

(Sketch). Let Φ be a PDL formula. In subformulae $\langle\alpha\rangle\varphi$ and $\langle\alpha\rangle^{-1}\varphi$ of Φ, view α as regular expression over finite alphabet $\{\,\mathsf{proc}, \mathsf{msg}, \{\varphi_1\}, \ldots, \{\varphi_n\}\,\}$ with local formulae φ_i (in Φ).

Model checking MSCs

Model checking MSCs versus PDL [Kern, 2009]

The following model-checking problem is decidable in polynomial time:

INPUT: MSC M, PDL-formula Φ

OUTPUT: does $M \models \Phi$?

Proof.

(Sketch). Let Φ be a PDL formula. In subformulae $\langle\alpha\rangle\varphi$ and $\langle\alpha\rangle^{-1}\varphi$ of Φ, view α as regular expression over finite alphabet $\{\,\mathsf{proc}, \mathsf{msg}, \{\varphi_1\}, \ldots, \{\varphi_n\}\,\}$ with local formulae φ_i (in Φ). Any such expression can be transformed into a corresponding finite automaton of linear size.

Model checking MSCs

Model checking MSCs versus PDL [Kern, 2009]

The following model-checking problem is decidable in polynomial time:

INPUT: MSC M, PDL-formula Φ

OUTPUT: does $M \models \Phi$?

Proof.

(Sketch). Let Φ be a PDL formula. In subformulae $\langle\alpha\rangle\varphi$ and $\langle\alpha\rangle^{-1}\varphi$ of Φ, view α as regular expression over finite alphabet $\{\, \mathsf{proc}, \mathsf{msg}, \{\varphi_1\}, \ldots, \{\varphi_n\} \,\}$ with local formulae φ_i (in Φ). Any such expression can be transformed into a corresponding finite automaton of linear size. We proceed by inductively labelling events of the given MSC with states of the finite automata.

Model checking MSCs

Model checking MSCs versus PDL [Kern, 2009]

The following model-checking problem is decidable in polynomial time:

INPUT: MSC M, PDL-formula Φ

OUTPUT: does $M \models \Phi$?

Proof.

(Sketch). Let Φ be a PDL formula. In subformulae $\langle\alpha\rangle\varphi$ and $\langle\alpha\rangle^{-1}\varphi$ of Φ, view α as regular expression over finite alphabet $\{ \textsf{proc}, \textsf{msg}, \{\varphi_1\}, \ldots, \{\varphi_n\} \}$ with local formulae φ_i (in Φ). Any such expression can be transformed into a corresponding finite automaton of linear size. We proceed by inductively labelling events of the given MSC with states of the finite automata. This state information is then used to discover whether or not an event of M satisfies a sub-formula $\langle\alpha\rangle\varphi$ and $\langle\alpha\rangle^{-1}\varphi$ which yields labellings in $\{ 0, 1 \}$.

Model checking MSCs

Model checking MSCs versus PDL [Kern, 2009]

The following model-checking problem is decidable in polynomial time:

INPUT: MSC M, PDL-formula Φ

OUTPUT: does $M \models \Phi$?

Proof.

(Sketch). Let Φ be a PDL formula. In subformulae $\langle\alpha\rangle\varphi$ and $\langle\alpha\rangle^{-1}\varphi$ of Φ, view α as regular expression over finite alphabet $\{\, \mathsf{proc}, \mathsf{msg}, \{\varphi_1\}, \ldots, \{\varphi_n\} \,\}$ with local formulae φ_i (in Φ). Any such expression can be transformed into a corresponding finite automaton of linear size. We proceed by inductively labelling events of the given MSC with states of the finite automata. This state information is then used to discover whether or not an event of M satisfies a sub-formula $\langle\alpha\rangle\varphi$ and $\langle\alpha\rangle^{-1}\varphi$ which yields labellings in $\{\,0, 1\,\}$. Boolean combinations and $\exists\varphi$ and $\forall\varphi$ are then handled in a straightforward manner.

Model checking MSCs

Model checking MSCs versus PDL [Kern, 2009]

The following model-checking problem is decidable in polynomial time:

INPUT: MSC M, PDL-formula Φ

OUTPUT: does $M \models \Phi$?

Proof.

(Sketch). Let Φ be a PDL formula. In subformulae $\langle\alpha\rangle\varphi$ and $\langle\alpha\rangle^{-1}\varphi$ of Φ, view α as regular expression over finite alphabet $\{\,\mathsf{proc}, \mathsf{msg}, \{\varphi_1\}, \ldots, \{\varphi_n\}\,\}$ with local formulae φ_i (in Φ). Any such expression can be transformed into a corresponding finite automaton of linear size. We proceed by inductively labelling events of the given MSC with states of the finite automata. This state information is then used to discover whether or not an event of M satisfies a sub-formula $\langle\alpha\rangle\varphi$ and $\langle\alpha\rangle^{-1}\varphi$ which yields labellings in $\{\,0, 1\,\}$. Boolean combinations and $\exists\varphi$ and $\forall\varphi$ are then handled in a straightforward manner. Time complexity: $\mathcal{O}(|E| \cdot |\Phi|^2)$ with $|E|$ is the number of events in M and $|\Phi|$ the length of Φ. □

PDL model checking algorithm for MSCs (1)

LOCAL FORMULA CHECK:

```
V = {0, .. , n-1}

boolean[] Sat(LocalFormula f) {
    boolean[] sat = new boolean[n];
    switch(f) {
    case Not(f1):
        boolean[] sat1 = Sat(f1);
        for (int i = 0; i < n; i++)
            sat[i] = !sat1[i];
        break;
    case Or(f1, f2):
        boolean[] sat1 = Sat(f1);
        boolean[] sat2 = Sat(f2);
        for (int i = 0; i < n; i++)
            sat[i] = sat1[i] || sat2[i];
        break;
    case Event(..):
        for (int i = 0; i < n; i++)
            sat[i] = (V[i].event.equals(f));
        break;
```

RWTH AACHEN UNIVERSITY

PDL model checking algorithm for MSCs (2)

```
    case <p1> f2:
        boolean[][] trans1 = Trans(p1);
        boolean[] sat2 = Sat(f2);
        for (int i = 0; i < n; i++) {
            sat[i] = false;
            for (int j = 0; j < n; j++)
                if(trans[i][j])
                sat[i] = sat2[j];
        }
        break;
    case <p1>^-1 f2:
        boolean[][] trans1 = TransBack(p1);
        boolean[] sat2 = Sat(f2);
        for (int i = 0; i < n; i++) {
            sat[i] = false;
            for (int j = 0; j < n; j++)
                if(trans[i][j])
                sat[i] = sat2[j];
        }
        break;
    }
}
```

PDL model checking algorithm for MSCs (3)

FORWARD PATH EXPRESSION CHECK:

```
boolean[][] Trans(PathFormula p) {
    boolean[][] trans = new boolean[n][n];
    switch(p) {
    case (p1; p2):
        boolean[][] trans1 = Trans(p1);
        boolean[][] trans2 = Trans(p2);
        for (int i = 0; i < n; i++)
            for (int k = 0; k < n; k++) {
                trans[i][k] = false;
                for (int j = 0; j < n; j++)
                    if(trans1[i][j] && trans1[j][k])
                        trans[i][k] = true;
            }
        break;
    case p1 + p2:
        boolean[][] trans1 = Trans(p1);
        boolean[][] trans2 = Trans(p2);
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                trans[i][j] = trans1[i][j] || trans2[i][j];
        break;
```

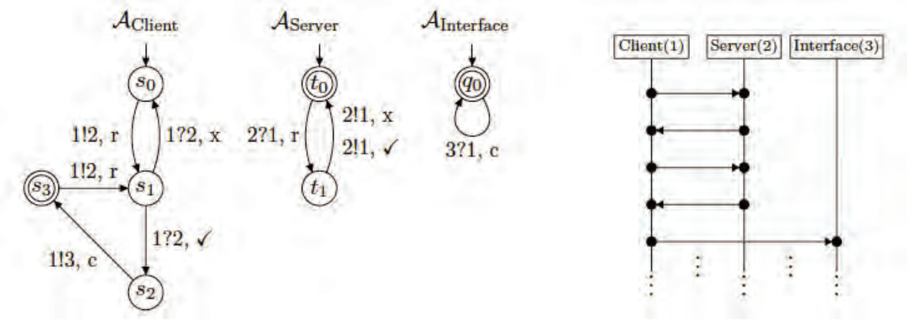
PDL model checking algorithm for MSCs (4)

```
    case p1*:
        boolean[][] trans1 = Trans(p1);
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                star[i][j] = (i==j);
        while (true) {
            for (int i = 0; i < n; i++)
                for (int j = 0; j < n; j++)
                    if (trans1[i][j])
                        for (int k = 0; k < n; k++)
                            if (!trans[i][k] && trans1[j][k]) {
                                trans[i][k] = true;
                                continue;
                            }
            break;
        }
        break;
    }
}
```

Communication finite-state machines

Let a CFM now be accepting if all its processes have reached a local accepting state and either halt there or visit a local accepting state infinitely often.

An example CFM and an infinite MSC accepted by it



Client-server interaction to get access to an interface. Accepting state is (s_3, t_0, q_0).

PDL formulas on CFMs

A CFM is accepting if all its processes have reached a local accepting state and reside their ad infinitum.

PDL formulas on CFMs

A CFM is accepting if all its processes have reached a local accepting state and reside their ad infinitum.

The language $\mathcal{L}(\mathcal{A})$ of CFM $\mathcal{A}$ is the set of MSCs that admit an accepting run.

PDL formulas on CFMs

A CFM is accepting if all its processes have reached a local accepting state and reside their ad infinitum.

The language $\mathcal{L}(\mathcal{A})$ of CFM $\mathcal{A}$ is the set of MSCs that admit an accepting run.

CFM versus PDL

A CFM $\mathcal{A}$ satisfies PDL-formula Φ, denoted $\mathcal{A} \models \Phi$, whenever for all MSCs M it holds: $M \in \mathcal{L}(\mathcal{A})$ if and only if $M \models \Phi$.

RWTH AACHEN UNIVERSITY

PDL formulas on CFMs

A CFM is accepting if all its processes have reached a local accepting state and reside their ad infinitum.

The language $\mathcal{L}(\mathcal{A})$ of CFM $\mathcal{A}$ is the set of MSCs that admit an accepting run.

CFM versus PDL

A CFM $\mathcal{A}$ satisfies PDL-formula Φ, denoted $\mathcal{A} \models \Phi$, whenever for all MSCs M it holds: $M \in \mathcal{L}(\mathcal{A})$ if and only if $M \models \Phi$.

The example CFM satisfies $\forall \, (P_1 \rightarrow (\langle \mathsf{proc}^*; \mathsf{msg}; \mathsf{proc}^*; \mathsf{msg} \rangle \, P_3)$ where for $i \in \mathcal{P}$, formula $P_i = \bigvee_{j \in \mathcal{P}, j \neq i} (!_{i,j} \vee ?_{i,j})$, i.e., $M, e \models P_i$ iff e occurs at process i. The PDL formula asserts that process 3 (Interface) can be "reached" from 1 (Client) by exactly two messages using an intermediate process in between.

PDL model checking problem

Model checking CFMs versus PDL

The following model-checking problem is undecidable:

INPUT: a CFM $\mathcal{A}$, PDL-formula Φ

OUTPUT: is there an MSC $M \in \mathcal{L}(\mathcal{A})$ with $M \models \Phi$?

PDL model checking problem

Model checking CFMs versus PDL

The following model-checking problem is undecidable:

INPUT: a CFM $\mathcal{A}$, PDL-formula Φ

OUTPUT: is there an MSC $M \in \mathcal{L}(\mathcal{A})$ with $M \models \Phi$?

Proof.

Follows immediately from the fact that the emptiness problem for CFMs is undecidable. By using the formula *true*, the above problem encodes the emptiness problem. □

PDL model checking problem

Model checking CFMs versus PDL

The following model-checking problem is undecidable:

INPUT: a CFM $\mathcal{A}$, PDL-formula Φ

OUTPUT: is there an MSC $M \in \mathcal{L}(\mathcal{A})$ with $M \models \Phi$?

Proof.

Follows immediately from the fact that the emptiness problem for CFMs is undecidable. By using the formula *true*, the above problem encodes the emptiness problem. □

To obtain decidable model-checking problems, we consider B-bounded MSCs.

Model checking CFMs versus PDL

Model checking CFMs versus PDL [Bollig *et. al*, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a CFM $\mathcal{A}$ and $B \in \mathbb{N}_{>0}$, PDL-formula Φ

OUTPUT: is there an $\exists B$-bounded MSC $M \in \mathcal{L}(\mathcal{A})$ with $M \models \Phi$?

Model checking CFMs versus PDL

Model checking CFMs versus PDL [Bollig *et. al*, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a CFM $\mathcal{A}$ and $B \in \mathbb{N}_{>0}$, PDL-formula Φ

OUTPUT: is there an $\exists B$-bounded MSC $M \in \mathcal{L}(\mathcal{A})$ with $M \models \Phi$?

Proof.

(Sketch).

Model checking CFMs versus PDL [Bollig *et. al*, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a CFM $\mathcal{A}$ and $B \in \mathbb{N}_{>0}$, PDL-formula Φ

OUTPUT: is there an $\exists B$-bounded MSC $M \in \mathcal{L}(\mathcal{A})$ with $M \models \Phi$?

Proof.

(Sketch). Every PDL formula Φ can effectively be translated into a CFM $\mathcal{A}_\Phi$ such that $\mathcal{A}_\Phi \models \Phi$.

Model checking CFMs versus PDL

Model checking CFMs versus PDL [Bollig *et. al*, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a CFM $\mathcal{A}$ and $B \in \mathbb{N}_{>0}$, PDL-formula Φ

OUTPUT: is there an $\exists B$-bounded MSC $M \in \mathcal{L}(\mathcal{A})$ with $M \models \Phi$?

Proof.

(Sketch). Every PDL formula Φ can effectively be translated into a CFM $\mathcal{A}_\Phi$ such that $\mathcal{A}_\Phi \models \Phi$. The details are out of the scope of this lecture.

Model checking CFMs versus PDL

Model checking CFMs versus PDL [Bollig *et. al*, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a CFM $\mathcal{A}$ and $B \in \mathbb{N}_{>0}$, PDL-formula Φ

OUTPUT: is there an $\exists B$-bounded MSC $M \in \mathcal{L}(\mathcal{A})$ with $M \models \Phi$?

Proof.

(Sketch). Every PDL formula Φ can effectively be translated into a CFM $\mathcal{A}_\Phi$ such that $\mathcal{A}_\Phi \models \Phi$. The details are out of the scope of this lecture. This synthesis step is independent of the channel bound size B (if any).

Model checking CFMs versus PDL

Model checking CFMs versus PDL [Bollig *et. al*, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a CFM $\mathcal{A}$ and $B \in \mathbb{N}_{>0}$, PDL-formula Φ

OUTPUT: is there an $\exists B$-bounded MSC $M \in \mathcal{L}(\mathcal{A})$ with $M \models \Phi$?

Proof.

(Sketch). Every PDL formula Φ can effectively be translated into a CFM $\mathcal{A}_\Phi$ such that $\mathcal{A}_\Phi \models \Phi$. The details are out of the scope of this lecture. This synthesis step is independent of the channel bound size B (if any). The size of $\mathcal{A}_\Phi$ is exponential in the length of Φ and the number of processes in $\mathcal{P}$.

Model checking CFMs versus PDL

Model checking CFMs versus PDL [Bollig *et. al*, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a CFM $\mathcal{A}$ and $B \in \mathbb{N}_{>0}$, PDL-formula Φ

OUTPUT: is there an $\exists B$-bounded MSC $M \in \mathcal{L}(\mathcal{A})$ with $M \models \Phi$?

Proof.

(Sketch). Every PDL formula Φ can effectively be translated into a CFM $\mathcal{A}_\Phi$ such that $\mathcal{A}_\Phi \models \Phi$. The details are out of the scope of this lecture. This synthesis step is independent of the channel bound size B (if any). The size of $\mathcal{A}_\Phi$ is exponential in the length of Φ and the number of processes in $\mathcal{P}$. Then construct a CFM accepting $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}_\Phi)$.

Model checking CFMs versus PDL

Model checking CFMs versus PDL [Bollig *et. al*, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a CFM $\mathcal{A}$ and $B \in \mathbb{N}_{>0}$, PDL-formula Φ

OUTPUT: is there an $\exists B$-bounded MSC $M \in \mathcal{L}(\mathcal{A})$ with $M \models \Phi$?

Proof.

(Sketch). Every PDL formula Φ can effectively be translated into a CFM $\mathcal{A}_\Phi$ such that $\mathcal{A}_\Phi \models \Phi$. The details are out of the scope of this lecture. This synthesis step is independent of the channel bound size B (if any). The size of $\mathcal{A}_\Phi$ is exponential in the length of Φ and the number of processes in $\mathcal{P}$. Then construct a CFM accepting $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}_\Phi)$. Decide whether the resulting CFM accepts some $\exists B$-bounded MSC.

Model checking CFMs versus PDL

Model checking CFMs versus PDL [Bollig *et. al*, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a CFM $\mathcal{A}$ and $B \in \mathbb{N}_{>0}$, PDL-formula Φ

OUTPUT: is there an $\exists B$-bounded MSC $M \in \mathcal{L}(\mathcal{A})$ with $M \models \Phi$?

Proof.

(Sketch). Every PDL formula Φ can effectively be translated into a CFM $\mathcal{A}_\Phi$ such that $\mathcal{A}_\Phi \models \Phi$. The details are out of the scope of this lecture. This synthesis step is independent of the channel bound size B (if any). The size of $\mathcal{A}_\Phi$ is exponential in the length of Φ and the number of processes in $\mathcal{P}$. Then construct a CFM accepting $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}_\Phi)$. Decide whether the resulting CFM accepts some $\exists B$-bounded MSC. This can all be done in polynomial space.

Model checking CFMs versus PDL

Model checking CFMs versus PDL [Bollig *et. al*, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a CFM $\mathcal{A}$ and $B \in \mathbb{N}_{>0}$, PDL-formula Φ

OUTPUT: is there an $\exists B$-bounded MSC $M \in \mathcal{L}(\mathcal{A})$ with $M \models \Phi$?

Proof.

(Sketch). Every PDL formula Φ can effectively be translated into a CFM $\mathcal{A}_\Phi$ such that $\mathcal{A}_\Phi \models \Phi$. The details are out of the scope of this lecture. This synthesis step is independent of the channel bound size B (if any). The size of $\mathcal{A}_\Phi$ is exponential in the length of Φ and the number of processes in $\mathcal{P}$. Then construct a CFM accepting $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}_\Phi)$. Decide whether the resulting CFM accepts some $\exists B$-bounded MSC. This can all be done in polynomial space. The PSPACE-hardness follows from the hardness of LTL model checking. □

Model checking MSGs versus PDL

Model checking MSGs versus PDL [Bollig et. al, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a MSG G and PDL-formula Φ

OUTPUT: is there an MSC $M \in \mathcal{L}(G)$ with $M \models \Phi$?

Model checking MSGs versus PDL

Model checking MSGs versus PDL [Bollig et. al, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a MSG G and PDL-formula Φ

OUTPUT: is there an MSC $M \in \mathcal{L}(G)$ with $M \models \Phi$?

Proof.

(Sketch.)

Model checking MSGs versus PDL

Model checking MSGs versus PDL [Bollig et. al, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a MSG G and PDL-formula Φ

OUTPUT: is there an MSC $M \in \mathcal{L}(G)$ with $M \models \Phi$?

Proof.

(Sketch.) For every vertex v, we can determine a linearization of the MSC $\lambda(v)$.

Model checking MSGs versus PDL

Model checking MSGs versus PDL [Bollig et. al, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a MSG G and PDL-formula Φ

OUTPUT: is there an MSC $M \in \mathcal{L}(G)$ with $M \models \Phi$?

Proof.

(Sketch.) For every vertex v, we can determine a linearization of the MSC $\lambda(v)$.

Construct a finite automaton $\mathcal{A}_G$ that accepts a linearization for every $M \in \mathcal{L}(G)$, and vice versa, each word accepted by $\mathcal{A}_G$ is a linearization of some $M \in \mathcal{L}(G)$.

Model checking MSGs versus PDL

Model checking MSGs versus PDL [Bollig et. al, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a MSG G and PDL-formula Φ

OUTPUT: is there an MSC $M \in \mathcal{L}(G)$ with $M \models \Phi$?

Proof.

(Sketch.) For every vertex v, we can determine a linearization of the MSC $\lambda(v)$. Construct a finite automaton $\mathcal{A}_G$ that accepts a linearization for every $M \in \mathcal{L}(G)$, and vice versa, each word accepted by $\mathcal{A}_G$ is a linearization of some $M \in \mathcal{L}(G)$. The size of $\mathcal{A}_G$ is linear in the size of G.

Model checking MSGs versus PDL

Model checking MSGs versus PDL [Bollig et. al, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a MSG G and PDL-formula Φ

OUTPUT: is there an MSC $M \in \mathcal{L}(G)$ with $M \models \Phi$?

Proof.

(Sketch.) For every vertex v, we can determine a linearization of the MSC $\lambda(v)$. Construct a finite automaton $\mathcal{A}_G$ that accepts a linearization for every $M \in \mathcal{L}(G)$, and vice versa, each word accepted by $\mathcal{A}_G$ is a linearization of some $M \in \mathcal{L}(G)$. The size of $\mathcal{A}_G$ is linear in the size of G. Construct a CFM $\mathcal{A}_\Phi$ for PDL-formula Φ with $M \in \mathcal{L}(\mathcal{A}_\Phi)$ iff $M \models \Phi$.

Model checking MSGs versus PDL

Model checking MSGs versus PDL [Bollig et. al, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a MSG G and PDL-formula Φ

OUTPUT: is there an MSC $M \in \mathcal{L}(G)$ with $M \models \Phi$?

Proof.

(Sketch.) For every vertex v, we can determine a linearization of the MSC $\lambda(v)$. Construct a finite automaton $\mathcal{A}_G$ that accepts a linearization for every $M \in \mathcal{L}(G)$, and vice versa, each word accepted by $\mathcal{A}_G$ is a linearization of some $M \in \mathcal{L}(G)$. The size of $\mathcal{A}_G$ is linear in the size of G. Construct a CFM $\mathcal{A}_\Phi$ for PDL-formula Φ with $M \in \mathcal{L}(\mathcal{A}_\Phi)$ iff $M \models \Phi$. Construct a transition system by running $\mathcal{A}_G$ and $\mathcal{A}_\Phi$ simultaneously.

Model checking MSGs versus PDL

Model checking MSGs versus PDL [Bollig et. al, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a MSG G and PDL-formula Φ

OUTPUT: is there an MSC $M \in \mathcal{L}(G)$ with $M \models \Phi$?

Proof.

(Sketch.) For every vertex v, we can determine a linearization of the MSC $\lambda(v)$. Construct a finite automaton $\mathcal{A}_G$ that accepts a linearization for every $M \in \mathcal{L}(G)$, and vice versa, each word accepted by $\mathcal{A}_G$ is a linearization of some $M \in \mathcal{L}(G)$. The size of $\mathcal{A}_G$ is linear in the size of G. Construct a CFM $\mathcal{A}_\Phi$ for PDL-formula Φ with $M \in \mathcal{L}(\mathcal{A}_\Phi)$ iff $M \models \Phi$. Construct a transition system by running $\mathcal{A}_G$ and $\mathcal{A}_\Phi$ simultaneously. This construction terminates as $\mathcal{A}_G$ only accepts linearizations that are B-bounded (as every linearization of MSG G is $\exists B$-bounded by definition).

Model checking MSGs versus PDL

Model checking MSGs versus PDL [Bollig et. al, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a MSG G and PDL-formula Φ

OUTPUT: is there an MSC $M \in \mathcal{L}(G)$ with $M \models \Phi$?

Proof.

(Sketch.) For every vertex v, we can determine a linearization of the MSC $\lambda(v)$. Construct a finite automaton $\mathcal{A}_G$ that accepts a linearization for every $M \in \mathcal{L}(G)$, and vice versa, each word accepted by $\mathcal{A}_G$ is a linearization of some $M \in \mathcal{L}(G)$. The size of $\mathcal{A}_G$ is linear in the size of G. Construct a CFM $\mathcal{A}_\Phi$ for PDL-formula Φ with $M \in \mathcal{L}(\mathcal{A}_\Phi)$ iff $M \models \Phi$. Construct a transition system by running $\mathcal{A}_G$ and $\mathcal{A}_\Phi$ simultaneously. This construction terminates as $\mathcal{A}_G$ only accepts linearizations that are B-bounded (as every linearization of MSG G is $\exists B$-bounded by definition). Deciding whether some simultaneous run is accepting can be done in polynomial space.

Model checking MSGs versus PDL

Model checking MSGs versus PDL [Bollig et. al, 2011]

The following model-checking problem is PSPACE-complete:

INPUT: a MSG G and PDL-formula Φ

OUTPUT: is there an MSC $M \in \mathcal{L}(G)$ with $M \models \Phi$?

Proof.

(Sketch.) For every vertex v, we can determine a linearization of the MSC $\lambda(v)$. Construct a finite automaton $\mathcal{A}_G$ that accepts a linearization for every $M \in \mathcal{L}(G)$, and vice versa, each word accepted by $\mathcal{A}_G$ is a linearization of some $M \in \mathcal{L}(G)$. The size of $\mathcal{A}_G$ is linear in the size of G. Construct a CFM $\mathcal{A}_\Phi$ for PDL-formula Φ with $M \in \mathcal{L}(\mathcal{A}_\Phi)$ iff $M \models \Phi$. Construct a transition system by running $\mathcal{A}_G$ and $\mathcal{A}_\Phi$ simultaneously. This construction terminates as $\mathcal{A}_G$ only accepts linearizations that are B-bounded (as every linearization of MSG G is $\exists B$-bounded by definition). Deciding whether some simultaneous run is accepting can be done in polynomial space. The PSPACE-hardness follows from the hardness of LTL model checking. □

Satisfiability problem for MSCs

Model checking MSCs versus PDL [Kern, 2009]

The following model-checking problem is decidable in polynomial time:

INPUT: MSC M, PDL-formula Φ

OUTPUT: does $M \models \Phi$?

Satisfiability problem for MSCs

Model checking MSCs versus PDL [Kern, 2009]

The following model-checking problem is decidable in polynomial time:

INPUT: MSC M, PDL-formula Φ

OUTPUT: does $M \models \Phi$?

MSC satisfiability for PDL [Bollig *et. al*, 2011]

The following satisfiability problem is undecidable:

INPUT: PDL-formula Φ

OUTPUT: is there an MSC M with $M \models \Phi$?

Other PDL decision problems

Theorem: [Alur *et al.*, 2001, Bollig *et al.*, 2007]

Let Φ be a PDL formula. Then:

1. The decision problem "does there exist a CFM $\mathcal{A}$ such that for any MSC $M \in \mathcal{L}(\mathcal{A})$ we have $M \models \Phi$" is undecidable.

Other PDL decision problems

Theorem: [Alur *et al.*, 2001, Bollig *et al.*, 2007]

Let Φ be a PDL formula. Then:

1. The decision problem "does there exist a CFM $\mathcal{A}$ such that for any MSC $M \in \mathcal{L}(\mathcal{A})$ we have $M \models \Phi$" is undecidable.
2. The decision problem "does there exist a CFM $\mathcal{A}$ such that for some $\exists B$-bounded MSC $M \in \mathcal{L}(\mathcal{A})$ we have $M \models \Phi$" is decidable in PSPACE.

Other PDL decision problems

Theorem: [Alur *et al.*, 2001, Bollig *et al.*, 2007]

Let Φ be a PDL formula. Then:

1. The decision problem "does there exist a CFM $\mathcal{A}$ such that for any MSC $M \in \mathcal{L}(\mathcal{A})$ we have $M \models \Phi$" is undecidable.
2. The decision problem "does there exist a CFM $\mathcal{A}$ such that for some $\exists B$-bounded MSC $M \in \mathcal{L}(\mathcal{A})$ we have $M \models \Phi$" is decidable in PSPACE.
3. The decision problem "for MSG G, is there an MSC $M \in \mathcal{L}(G)$ such that $M \models \Phi$" is NP-complete.