Theoretical Foundations of the UML

Lecture [redacted] 16: A Logic for MSCs (Part 2)

Joost-Pieter Katoen

Lehrstuhl für Informatik 2
Software Modeling and Verification Group

`moves.rwth-aachen.de/teaching/ss-20/fuml/`

June 16, 2020

Outline

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

Propositional
Dynamic
Logic

MSC M PDL-formula Φ

$M \models \Phi$?

MSG g PDL-formula Φ

$\forall M \in L(g).\ M \models \Phi$?

Φ $\exists$ MSC $M.\ M \models \Phi$?

Overview

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:

!(1,2,a) ?(2,3,b) forward

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

backward

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

Definition (Derived operators)

$\langle\alpha\rangle\varphi$

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

α is a regular expressions describes the possible admitted ways to navigate through a MSC

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:

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

Definition (Syntax of path expressions)

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

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

local formula

vertically / downwards backwards

$\langle proc \rangle$ $\langle proc \rangle^{-1}$

horizontally

$\langle msg \rangle$ left2right

$\langle msg \rangle^{-1}$ right2left

PDL formulas

does the MSC contain an event satisfying φ

do all events in the MSC satisfy φ

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
- 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

$\equiv \exists \neg \langle\alpha\rangle \neg \varphi$

Semantics of PDL formulas

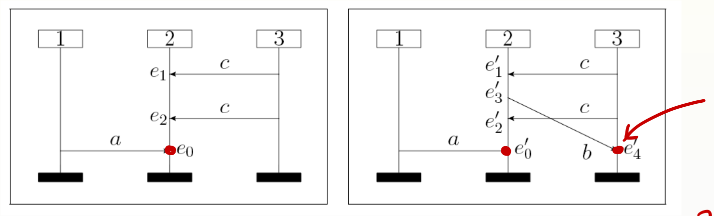
Definition (Semantics of PDL formulas)

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.

- $M \models \exists\varphi$ iff $\exists e \in E.\, M, e \models \varphi$
- $M \models \forall\varphi$ iff $\forall e \in E.\, M, e \models \varphi$
- $M \models \Phi_1 \wedge \Phi_2$ iff $M \models \Phi_1$ and $M \models \Phi_2$
- $M \models \Phi_1 \vee \Phi_2$ iff $M \models \Phi_1$ or $M \models \Phi_2$

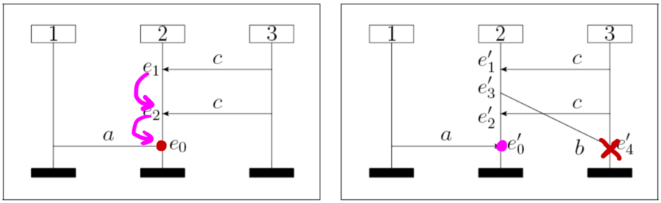
Example (1)



$e'_4 \models ?(2,1,a)$

- 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.

p = proc
m = msg

$$M \models \forall \langle (p+m)^* \rangle \underbrace{([p]\,\text{false} \wedge ?(2,1,a))}_{\varphi}$$

iff (* semantics of PDL formulas *)

$$\forall e \in E.\ \left(e \models \langle (p+m)^* \rangle ([p]\,\text{false} \wedge ?(2,1,a)) \right)$$

set of events in M

iff (* semantics of local formulas *)

$$\forall e \in E.\ \left(\exists n \in \mathbb{N}.\ e \models \langle (p+m) \rangle^n ([p]\,\text{false} \wedge ?(2,1,a)) \right)$$

intuitive: for every event e in the MSG, there exists an event e' such that $\boxed{e \prec^* e'}$ and

$$\underbrace{e' \models [p]\,\text{false} \wedge ?(2,1,a)}$$

e' has no successors at its process and it is labeled with ?(2,1,a)

$$\left\{ \begin{array}{l} \neg(\exists e''.\ e' \lessdot_p e'') \text{ and} \\ \ell(e') = ?(2,1,a) \end{array} \right. \quad (*)$$

M_{left} : Φ holds

$\hookrightarrow \forall \langle (p+m)^* \rangle ([p] \text{false} \wedge ?(2,1,a))$

take $e' = e_0$

$e_0 \models (*)$, and $\forall e \in E . e <^* e_0$

e.g. $e_1 \models \langle p \rangle \langle p \rangle e_0$ $(n=2)$

and similar for all other events

in M_{left}. Thus, $M_{left} \models \Phi$.

M_{right} : $e_0' \models [P] \text{false} \wedge ?(1,2,a)$

and is the only event in M_{right}

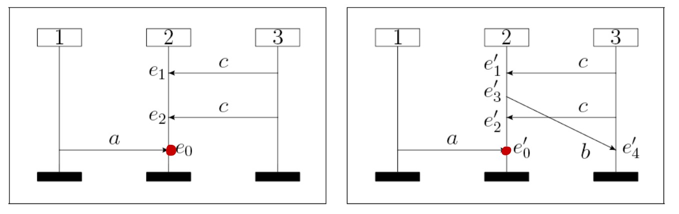
satisfying this formula

$\forall e \in E_{right} .$ $e <^* e_0'$

this does not hold as $e_4' <^* e_0'$.

Thus $M_{right} \not\models \Phi$.

Example (2)



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

$M \models \exists\,([p]\,\text{false} \wedge\ ?(2,1,a))$

iff $\exists e \in E.\ \left(e \models [p]\,\text{false} \wedge\ ?(2,1,a)\right)$

iff $\exists e \in E\ \left(e \models [p]\,\text{false} \text{ and } e \models\ ?(2,1,a)\right)$

iff $\exists e \in E\ \Big((\exists e' \in E.\ e <\!\cdot_p e' \wedge e' \models \text{false})$

$\quad$ and $\ell(e) =\ ?(2,1,a)\Big)$

iff $\exists e \in E\ \left(\neg\,(\exists e' \in E.\ e <\!\cdot_p e') \text{ and } \ell(e) =\ ?(2,1,a)\right)$

$M_{left} \models \Phi$ since $\ell(e_0) =\ ?(2,1,a)$

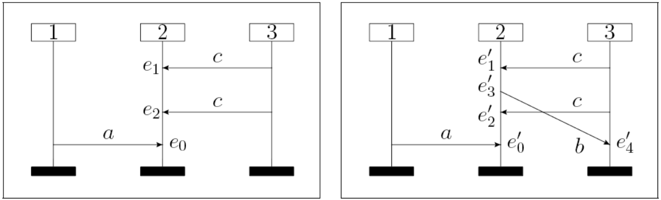
and e_0 has no successors

at its process

$M_{right} \models \Phi$ in a similar way using

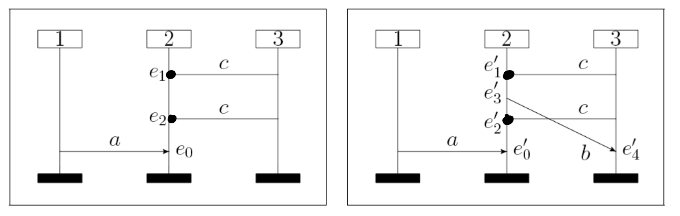
$e = e_0'$

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.

- $\forall\, ([\{\, ?(2, 3, c)\, \}; \mathsf{proc}; \{\, ?(2, 3, c)\, \}]\, \mathit{false})$ No. Yes.

?(2,3,c)

?(2,3,c)

$M \models \forall [\{?(2,3,c)\} ; p ; \{?(2,3,c)\}]$ false

iff $\forall e \in E.$ $e \models [\ldots\ldots]$ false

iff (* use that $[\alpha]\varphi \equiv \neg\langle\alpha\rangle\neg\varphi$ *)

$\forall e \in E.$ $e \models \neg \langle \ldots \rangle \underbrace{\neg \text{false}}_{\text{true}}$

iff $\forall e \in E.$ not ($e \models \langle \ldots \rangle$ true)

iff $\forall e \in E.$ not ($e \models \langle\{?(2,3,c)\}\rangle\langle p\rangle\langle\{?(2,3,c)\}\rangle$ true)

iff $\forall e \in E.$ not ($\ell(e) = ?(2,3,c) \wedge e \models \langle p \rangle \langle \ldots \rangle$ true)

iff $\forall e \in E.$ not($\underline{\ell(e) = ?(2,3,c)}$ and

$\uparrow$ $\exists e' \in E.$ $e <_p e'$ and $\ell(e') = ?(2,3,c)$)

$M_{left} \not\models \Phi$ take $e = e_1$ and $e' = e_2$

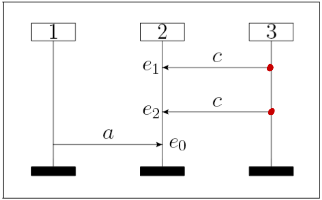
e_1 and e_2 violate the above formula

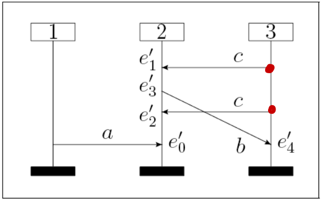
$M_{right} \models \Phi$ two cases $e = e_1'$ and $e = e_2'$

$e_1' <_p e_3'$ but $\ell(e_3') \neq ?(2,3,c)$

$e_2' <_p e_0'$ but $\ell(e_0') \neq ?(2,3,c)$

Example (4)





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

Abbreviations (auxiliary formulas)

$$!_{1,j} = \bigvee_{a \in C} \underbrace{!(1,j,a)}_{\text{local formula}}$$

$a \in C$ ↓ message contents

send action at process 1 to process j

$$\bigvee_{p \in P} !_{1,p}$$

send action at process 1

$$?_{1,j} = \bigvee_{a \in C} ?(1,j,a)$$

receive some message from j at process 1

$$P_1 = \bigvee_{j \in P,\, j \neq 1} \left(\underline{!_{1,j}} \vee \underline{?_{1,j}} \right)$$

action at process 1

$e \models P_1$ iff e occurs at process 1

Path expression α asserting that a certain event at some process j happens an even number of times

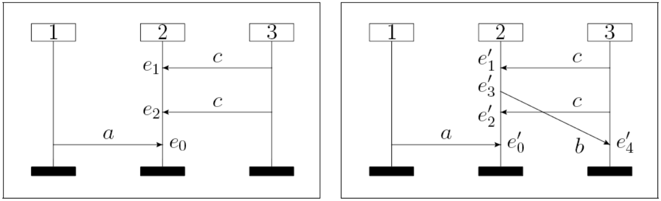
$\to$ 0, 2, 4, 6, ...

ψ (local formula)

$$\alpha = \left(\underbrace{\quad\quad}_{\text{no event satisfying } \psi \text{ occurs}} \{\psi\} \underbrace{\quad\quad}_{\text{no event satisfying } \psi} \{\psi\} \underbrace{\quad\quad}_{\text{no event satisfying } \psi} \right)^*$$

$(\{\neg\psi\} ; proc)^*$; $proc$; $(\{\neg\psi\} ; proc)^*$

Example (4)



- The number of send events at process 3 is odd. No. No.
- See next slide for a PDL-formula for a similar property.

Example

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

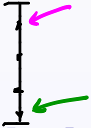
$$\forall\Big(\underbrace{[\mathsf{proc}]^{-1}\, \mathit{false} \wedge P_1}_{\text{minimal event on process 1}} \;\rightarrow\; \langle \alpha \rangle \; \underbrace{[\mathsf{proc}]\, \mathit{false} \wedge P_1}_{\text{maximal event on process}} \Big) \qquad (*)$$

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:

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

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

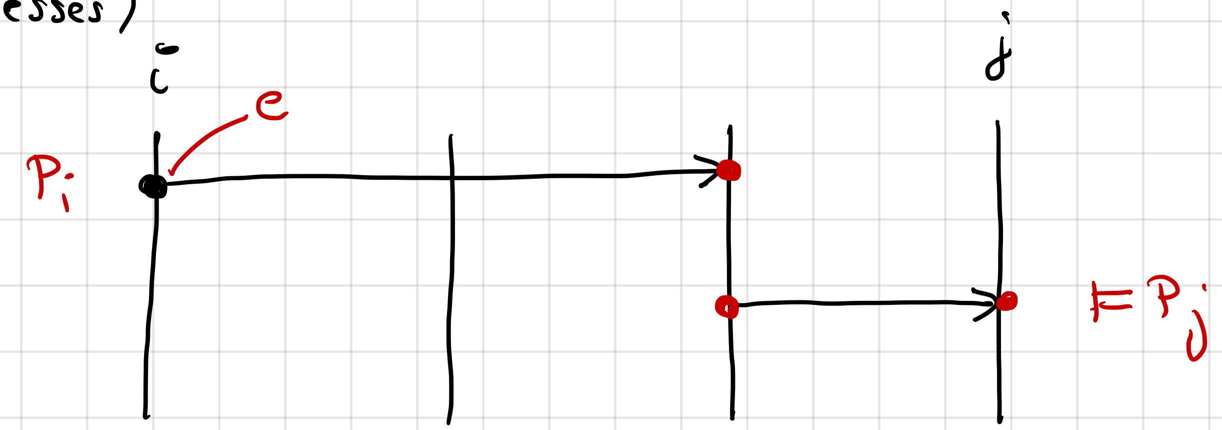


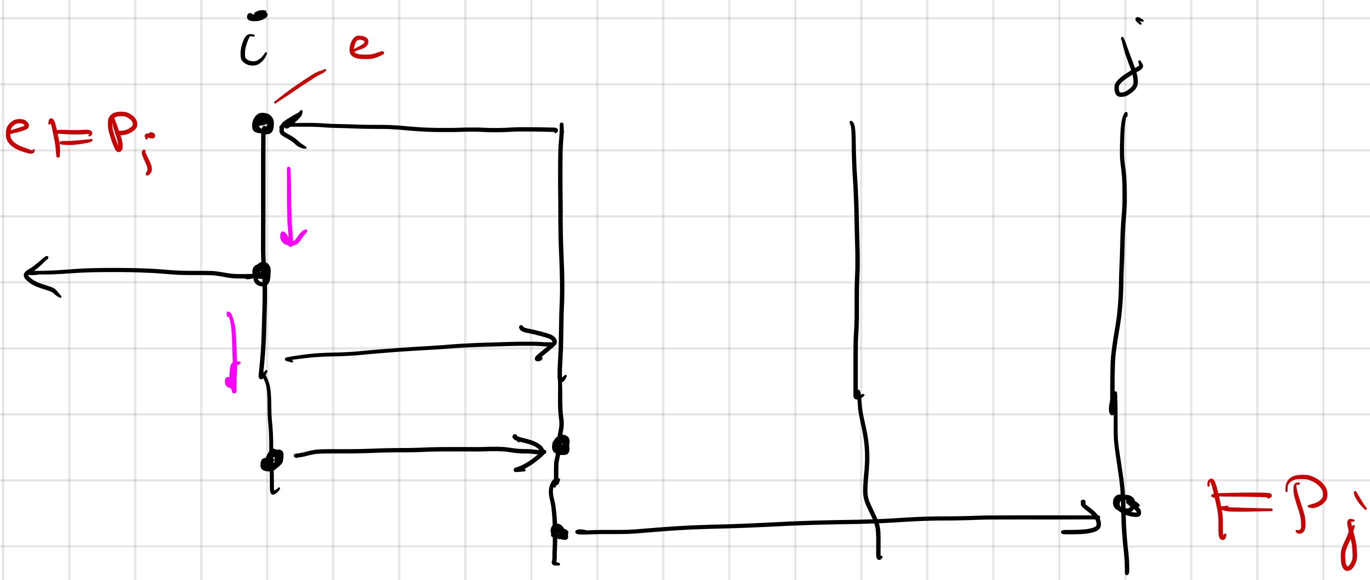
— = no $!_1$ event occurs

let $i \neq j$

$$\forall \left(P_i \longrightarrow \left(\langle \underline{proc^*}; msg; \underline{proc^*}; \underline{msg} \rangle P_j \right)\right)$$

expresses that process i can "reach" process j by exactly two messages (using intermediate processes)





Overview

given a MSC M
+ PDL-formula Φ
Does $M \models \Phi$?

PDL-formula Φ

Does there exist a MSC M, $M \models \Phi$?

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. 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 $\underline{\langle\alpha\rangle}\varphi$ and $\underline{\langle\alpha\rangle^{-1}\varphi}$ which yields labellings in $\underbrace{\{\,0, 1\,\}}_{\mathbb{B}}$.

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. $\square$

PDL model checking algorithm for MSCs (1)

LOCAL FORMULA CHECK:

```
V = {0, .. , n-1}                                   // set of events

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;
```

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;
    }
}
```

$\langle \alpha \rangle \varphi$

$\langle \alpha \rangle^{-1} \varphi$

trans [i] [j] = true iff $(e_i, e_j) \models p$

FORWARD PATH EXPRESSION CHECK:

$(e, e') \models p$

```
boolean[][] Trans(PathFormula p) {
    boolean[][] trans = new boolean[n][n];
    switch(p) {
    case (p1; p2):                                  // concatenation
        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:                                   // choice
        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)

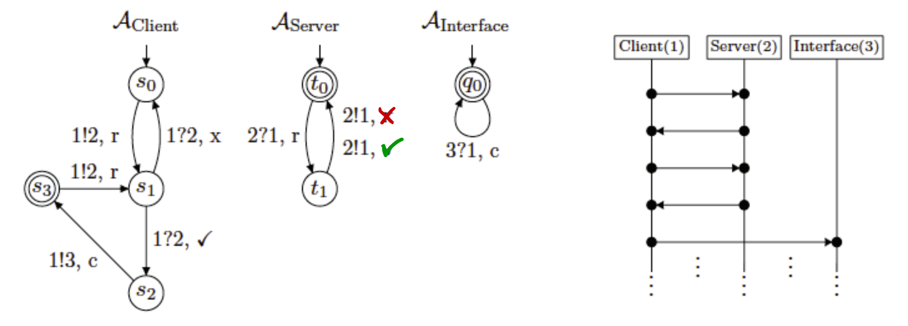
// Kleene star

```
    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.

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 $\forall\,(P_1 \rightarrow (\underline{\langle \mathsf{proc}^*; \mathsf{msg}; \mathsf{proc}^*; \mathsf{msg}\rangle\, P_3}))$ 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 $M \models \Phi$?

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. $\square$

$\forall$ true

$\exists$ true

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 [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 $\mathcal{A}_\Phi \models \Phi$. Construction is involved.

$\forall \Phi.\ \mathcal{M} = \{ M \in \mathbb{M} \mid M \models \Phi \}$ can be accepted by a CFM A such that $L(A) = \mathcal{M}$.

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 $\mathcal{P}$.

$\Phi \rightsquigarrow$ CFM A_Φ $\quad |A_\Phi| \in O(2^{|\Phi|})$

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 $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}_\Phi)$.

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. □

lecture Model Checking ← WiSe 2020/21

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. $\square$

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.
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.

PDL verification problems

MSC M PDL-formula Φ	$M \models \Phi$?	decidable in P
CFM $\mathcal{A}$ PDL-formula Φ	$\exists M \in L(\mathcal{A}).\ M \models \Phi$?	undecidable
CFM $\mathcal{A}$ PDL-formula Φ bound $B \in \mathbb{N}_{>0}$	$\exists M \in L(\mathcal{A}).$ $M \models \Phi$ and M is $\exists B$-bounded ?	decidable (PSPACE-complete)
MSG $\mathcal{G}$ PDL-formula Φ	$\exists M \in L(\mathcal{G}).\ M \models \Phi$?	decidable (PSPACE-complete)

PDL satisfiability problems

PDL-formula Φ	$\exists M \in \mathbb{M}.\ M \models \Phi$?	undecidable
PDL-formula Φ	$\exists$ CFM $\mathcal{A}$ such that $\forall M \in L(\mathcal{A}).\ M \models \Phi$?	undecidable
PDL-formula Φ bound $B \in \mathbb{N}$	$\exists$ CFM $\mathcal{A}$ such that $\exists M \in L(\mathcal{A}).\ M$ is $\exists B$-bounded $M \models \Phi$?	decidable [in PSPACE]

For logic-interested people:

(1) PDL $\subsetneq$ $\exists$ MSO-logic

monadic second order-logic

$\exists x \; \forall y$

$\exists X \; \forall Y \; \exists x \in X.$

(2) Extending PDL with intersection yields a logic that is more expressive than CFMs.

$\langle \alpha_1 \cap \alpha_2 \rangle \varphi$ "there exist two paths described by α_1 and α_2 resp. that both lead to an event satisfying φ"

Then [Bollig et al., 2010] showed:

$\exists \varphi$ such that $\{ M \in \mathbb{M} \mid M \models \varphi \}$ cannot be accepted by a CFM.

in extended PDL

③ PDL supports "forward" navigation

$$\langle \alpha \rangle \varphi$$

and "backward" navigation $\langle \alpha \rangle^{-1} \varphi$.

PDL does not allow to mix "forward" and "backward" in a single formula

e.g. $proc^* ; msg^{-1} ; proc$ is not a syntactically admitted formula.

④ The temporal logic formula φ until ψ, i.e. φ holds at all events until an event satisfying ψ is "reached" can be expressed as PDL-formula

$$\langle \underbrace{(\underbrace{\{\varphi\}} ; \underbrace{(proc + msg)})^*} \rangle \psi$$