Model-Checking Large Finite State Systems and Beyond

Luboš Brim & Mojmír Křetínský
Masaryk University Brno

Part II (Beyond FS): Infinite state systems

Mojmír Křetínský

A joint work with Vojtěch Řehák and Jan Strejček

Motivation

- modelling (some features of) current SW, e.g. programs with unbounded control structure (unbounded depth of recursive call, dynamic creation of concurrent process) and their synchronization

$$\Downarrow$$

need of infinite-state systems

- examples of standard formalisms:
  - process algebras: **BPA, BPP, PA**
  - **Petri nets (PN)**
  - **pushdown systems (PDA), recursive state machines**
  - **process rewrite systems (PRS)**
- **Goal:** find a balance between **expressiveness** and **decidability**
- we study extensions of PRS improving expressiveness and keeping decidability

Outline

- **Process rewrite system (PRS) and program modeling**
  - rewriting concepts in modeling
  - process rewrite system (PRS), hierarchy
  - subclasses of PRS and program modeling
- **Extensions of PRS and their expressiveness**
  - state extended PRS (sePRS)
  - **weakly extended PRS (wPRS)**
- **Decidability results and their applications**
  - **Reachability problem is decidable for wPRS**
  - **Reachability Hennessy-Milner property is decidable for wPRS**
  - **Model checking problem is decidable for wPRS and Lamport logic**

Process rewrite system (PRS)

- rewriting concepts in modeling
- process rewrite system (PRS) [Mayr98]
- subclasses of PRS and program modeling

PRS - An Intuitive Introduction (FS)

Rewrite systems (e.g. grammars) and LTS (e.g. automata state graphs)

(Process) rewrite systems as generators of process **behaviours** (Labelled transition systems)

interpret grammar rules (generating strings) as rules for generating behaviours.

Type 3 rules

- a rule $A \longrightarrow aB$ becomes a rule

$$A \xrightarrow{a} B$$

  A process A **can** **perform an action** a and **become** a process B

- a rule $A \longrightarrow a$ becomes a rule

$$A \xrightarrow{a} \varepsilon$$

  A process A **can** **perform an action** a and **terminate**

- can model nondeterministic finite state systems

PRS - An Intuitive Introduction: context-free rules

A context–free rule in GNF $A \longrightarrow aBC$ should become $A \xrightarrow{a} BC$

How to interpret concatenation (juxtaposition) BC ?

To model **sequential** and/or **parallel** compositions of systems we make use of

- sequential operator '$.$' (rather than Unix like ';', e.g. xterm; xterm;)
  '$.$' is associative

  and

- parallel operator '$\|$' (rather than Unix like '&', e.g. xterm& xterm&)
  '$\|$' is associative and commutative

PRS - An Intuitive Introduction: context-free

A sequential (context-free) rule

$$A \xrightarrow{a} B.C$$

- a process A can perform an action a and become a sequential composition of B and C
- in a state $B.C$ no C–rule is applicable (i.e. C cannot move) until B does not terminate (left derivations only – prefix rewriting)
- we can model recursive procedures

A parallel (context-free) rule

$$A \xrightarrow{a} B||C$$

- a process A can perform an action a and become a parallel composition of B and C
- in a state $B||C$ both C–rules and B–rules can be applied
- can model dynamic creation of (asynchronous) parallel processes

PRS - An Intuitive Introduction: context-free

Type 2 (context-free) sequential and parallel rules:

$$A \xrightarrow{a} (B \| C).D$$

cobegin – coend section, fork – join

PRS - An Intuitive Introduction: other rules

Type 0 ("context-sensitive") rules:

- parallel composition only (multiset rewriting)

$$A||B \xrightarrow{a} C||D$$

communication and synchronization

- sequential composition only (prefix rewriting)

$$C.D \xrightarrow{a} E$$

value passing, recursive calls returning values over finite data domains

- other combinations are useful as well

Rewriting models (of control flow graphs) of programs

Programs with

- procedures/methods and recursion, and/or
- concurrency and communication (processes/threads, cobegin-coend sections)

are "compiled" (abstracted) into simpler and formal models.

We use rewriting concepts, i.e. we model:

- program states as terms (states of LTS),
- program instructions as term-rewriting rules (generate transitions in LTS), and
- program executions as sequences of rewriting steps (paths in LTS).

Process Terms

Process terms are expressions of the form

$$t \;::=\; \varepsilon \;\mid\; X \;\mid\; t.t \;\mid\; t\|t$$

where

- ε is an **empty term**
- X ranges over set $\{A, B, C, \ldots\}$ of **process constants**
- "$.$" is associative **sequential** operator
- "$\|$" is associative and commutative **parallel** operator

e.g.

$$\varepsilon\|A = A\|\varepsilon = A = A.\varepsilon = \varepsilon.A$$

$$A\|(B\|C) \;=\; (A\|B)\|C \;=\; C\|(A\|B)$$

$$A.(B.C) \;=\; (A.B).C \;\neq\; C.(A.B)$$

Process Rewrite System (PRS)

PRS – a finite set of rewrite rules and an initial term

$$R = \{ \; A.B \overset{a}{\hookrightarrow} C\|D \;,\quad B \overset{b}{\hookrightarrow} A.B$$
$$C \overset{c}{\hookrightarrow} \varepsilon \;,\quad E \overset{d}{\hookrightarrow} F \; \}$$

LTS is induced by R starting in the initial term (state) $B\|E$

$$\begin{array}{ccccccc}
\to B\|E & \xrightarrow{b} & (A.B)\|E & \xrightarrow{a} & (C\|D)\|E & \xrightarrow{c} & \cdots \\
\downarrow d & & \downarrow d & & \downarrow d & & \\
B\|F & \xrightarrow{b} & (A.B)\|F & \xrightarrow{a} & (C\|D)\|F & \xrightarrow{c} & \cdots
\end{array}$$

Classes of Process Terms

We define these classes of process terms:

"1" process constants, e.g. A

"S" only sequential composition, e.g. $A.\mathrm{B}.\mathrm{C}$

"P" only parallel composition, e.g. $A\|\mathrm{B}\|\mathrm{C}$

"G" general terms, e.g. $A.(\mathrm{B}\|(\mathrm{C}.\mathrm{D}))$

Classes of Process Rewrite Systems

(1,1)-PRS class . . . . . . Finite-state Systems

$$R = \left\{ \begin{array}{llll} A \xhookrightarrow{a} B, & & B \xhookrightarrow{b} \varepsilon, & \\ A \xhookrightarrow{d} C, & & C \xhookrightarrow{c} \varepsilon & \end{array} \right\}$$

$$\begin{tikzcd} {} \arrow[r] & A \arrow[r, "a"] \arrow[d, "d"] & B \arrow[d, "b"] \\ & C \arrow[r, "c"] & \varepsilon \end{tikzcd}$$

Classes of Process Rewrite Systems

(1,P)-PRS class . . . . . . Basic Parallel Processes

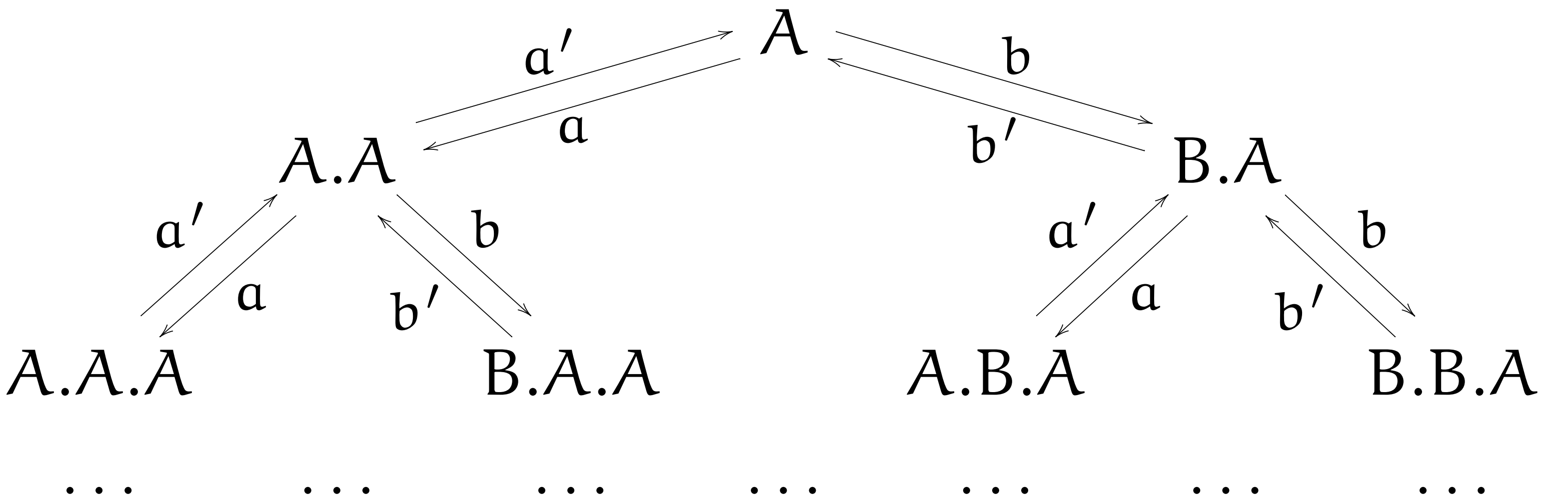
$$R = \{\ A \overset{a}{\hookrightarrow} A\|A\ ,\qquad A \overset{a'}{\hookrightarrow} \varepsilon\ ,\qquad B \overset{b}{\hookrightarrow} B\|B\ ,\qquad B \overset{b'}{\hookrightarrow} \varepsilon\ \}$$

$$\begin{array}{ccccccc}
\cdots & \underset{a'}{\overset{a}{\rightleftarrows}} & A\|A\|B & \underset{a'}{\overset{a}{\rightleftarrows}} & A\|A\|A\|B & \underset{a'}{\overset{a}{\rightleftarrows}} & \cdots \\
 & & b'\uparrow\downarrow b & & b'\uparrow\downarrow b & & \\
\cdots & \underset{a'}{\overset{a}{\rightleftarrows}} & A\|A\|B\|B & \underset{a'}{\overset{a}{\rightleftarrows}} & A\|A\|A\|B\|B & \underset{a'}{\overset{a}{\rightleftarrows}} & \cdots \\
 & & b'\uparrow\downarrow b & & b'\uparrow\downarrow b & & \\
\cdots & & \cdots & & \cdots & & \cdots
\end{array}$$

Classes of Process Rewrite Systems

(1,S)-PRS class ...... Basic Process Algebra

$$
\begin{aligned}
R = \{\ & A \overset{a}{\hookrightarrow} A.A\ , && B \overset{a}{\hookrightarrow} A.B\ , && A \overset{a'}{\hookrightarrow} \varepsilon\ , \\
& B \overset{b}{\hookrightarrow} B.B\ , && A \overset{b}{\hookrightarrow} B.A\ , && B \overset{b'}{\hookrightarrow} \varepsilon\ \}
\end{aligned}
$$



Classes of Process Rewrite Systems

(1,G)-PRS class . . . . . . Process Algebra

$$R = \{\ A \overset{a}{\hookrightarrow} A.A\ ,\qquad B \overset{a}{\hookrightarrow} A \| (B.C)\ , \ldots\}$$

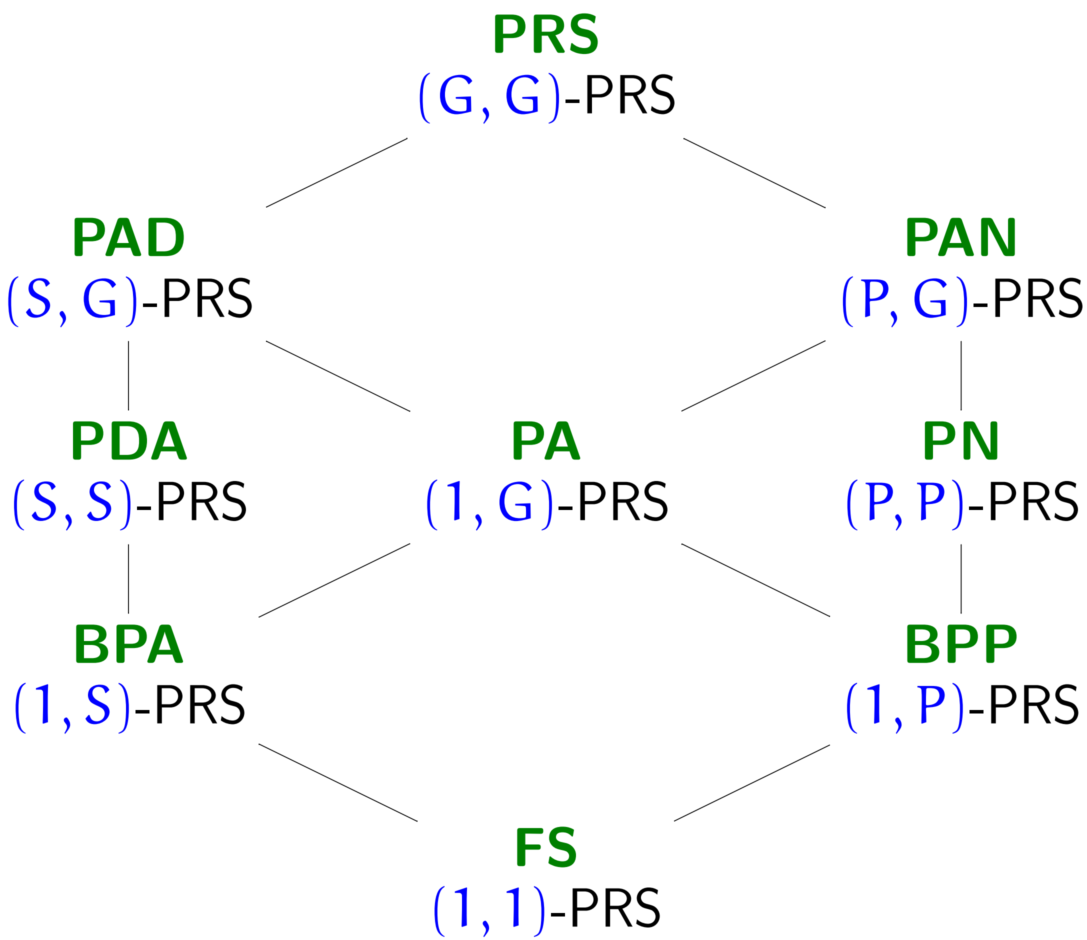
(S,S)-PRS class . . . . . . PDA

$$R = \{\ A.C \overset{a}{\hookrightarrow} D\ ,\qquad B.A.D \overset{a}{\hookrightarrow} B.C\ , \ldots\}$$

(P,P)-PRS class . . . . . . Petri Nets

$$R = \{\ A \| B \overset{a}{\hookrightarrow} C \| B\ ,\qquad B \overset{a}{\hookrightarrow} A \| C\ , \ldots\}$$

PRS hierarchy, models



is strict w.r.t. strong bisimulation equivalence

Extensions of PRS and their Expressiveness

- state extended PRS (sePRS),
- **weakly extended PRS (wPRS)**

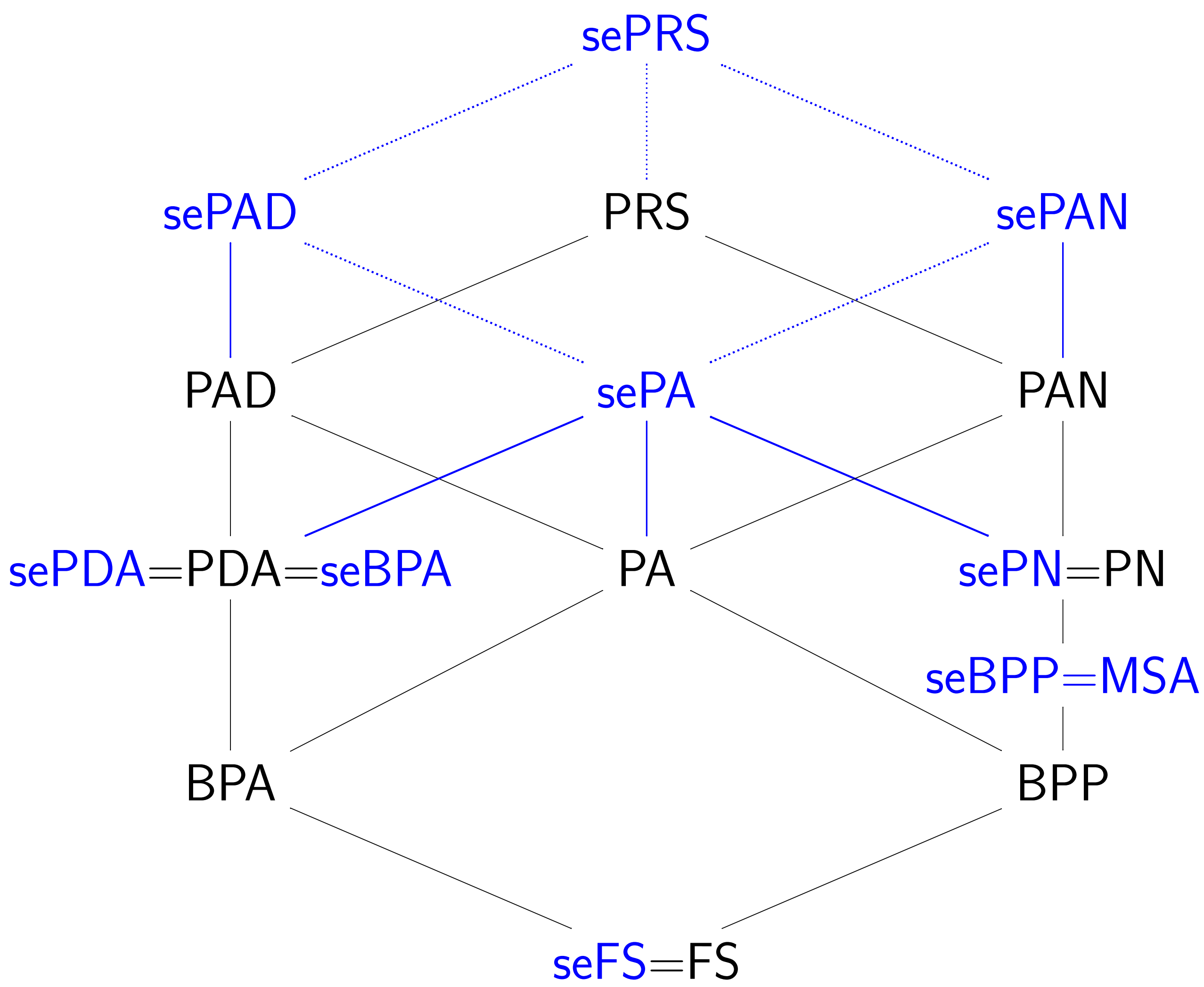
State Extended PRS (sePRS)

sePRS $=$ PRS $+$ finite-state control unit

e.g.
PDA $=$ BPA $+$ finite-state control unit

$$
\begin{aligned}
R = \{ \; & pA \overset{a}{\hookrightarrow} pA.A \;, && pA \overset{a'}{\hookrightarrow} p\varepsilon \;, \\
& pB \overset{b}{\hookrightarrow} qB.B \;, && qB \overset{b'}{\hookrightarrow} pB.B \; \}
\end{aligned}
$$

State Extended PRS-hierarchy



Motivation for Weak State Extension

4 (out of 5) new sePRS classes have a full Turing-power

$\Downarrow$

sePRS are too strong

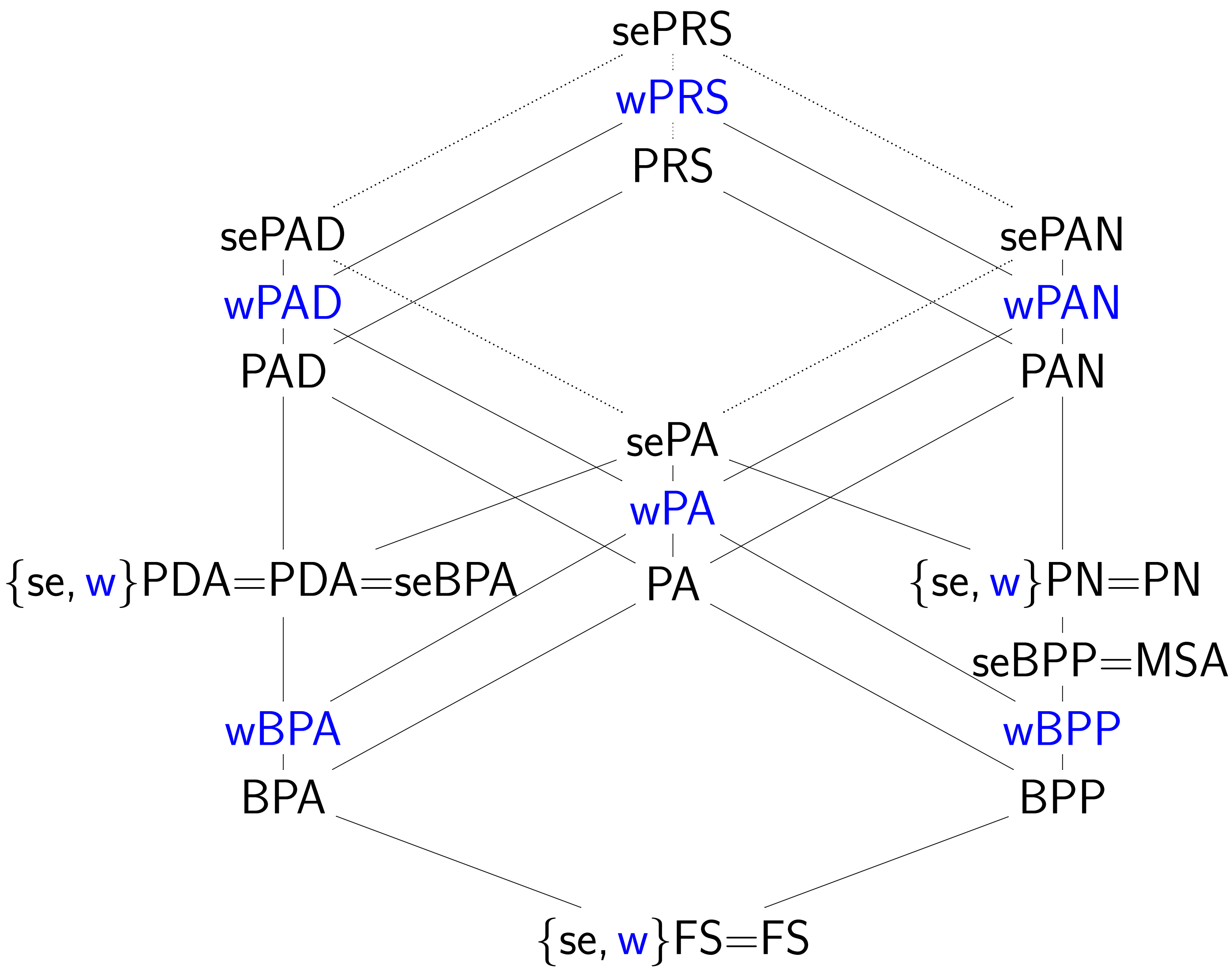
$\Downarrow$

weakly extended PRS (wPRS) [Infinity 2003]

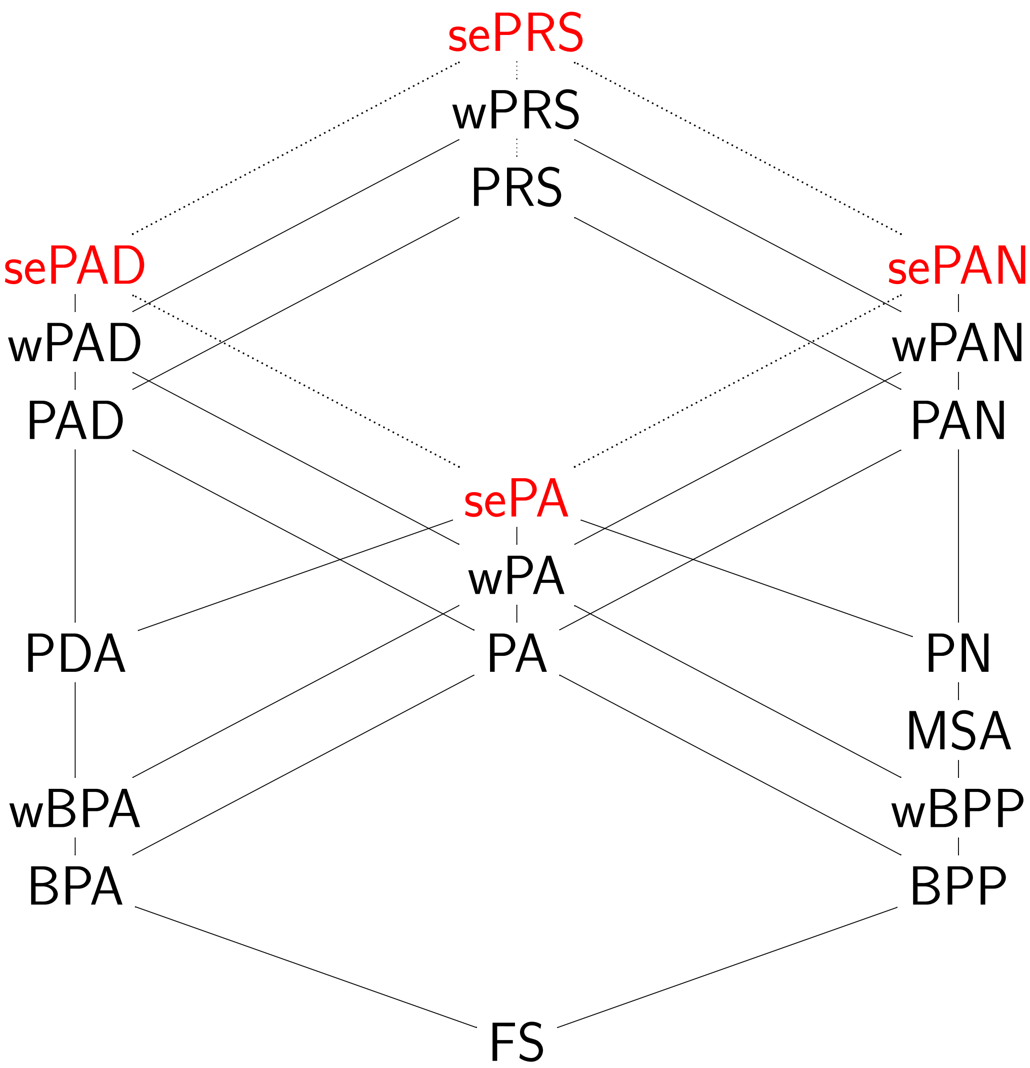
1-weak (or very weak) restriction from the automata theory
(i.e. no cycle except self-loops)

1-weak restriction: There is a partial ordering on states of the finite-state unit such that the transition relation respects the ordering

Extended PRS-hierarchy



Turing Powerfull Classes

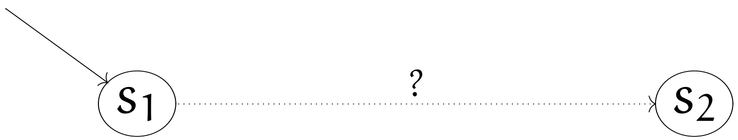


Reachability Problem

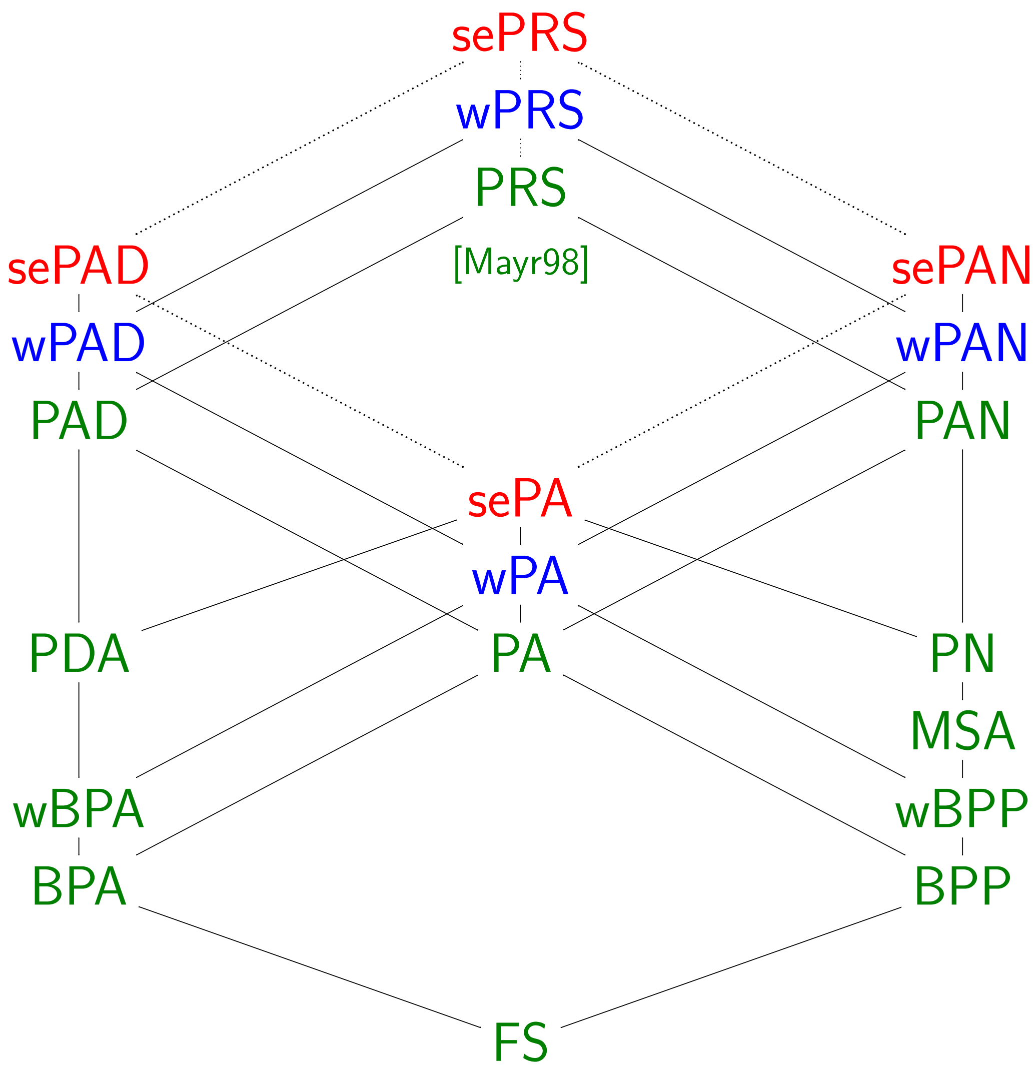
Instance: (α, β)-(se-,w-)PRS system and two of its states s_1, s_2

Question: Is the state s_2 reachable from s_1

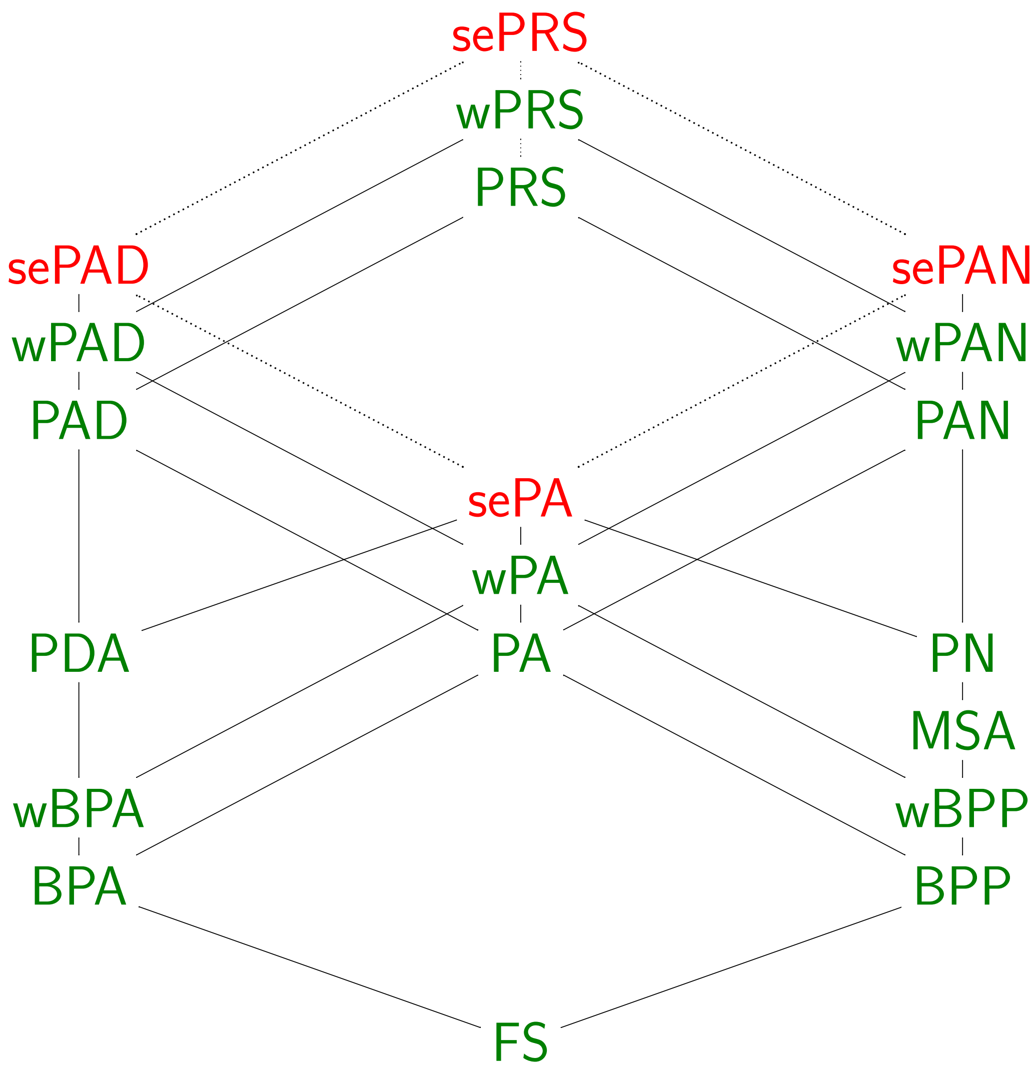
i.e. $s_1 \xrightarrow{*} s_2$?



Reachability Problem



Reachability Problem [CONCUR'04]



Reachability problem

Theorem [Mayr1998]: Reachability problem is decidable for PRS.

Theorem: Reachability problem is decidable for wPRS.

Applications:

- **model checking** some of safety properties
- **reachability** for Hüttel and Srba's **replicative variant** of Dolev and Yao's **ping-pong protocols** [Hüttel-Srba05]
- **weak trace non-equivalence** is semi-decidable for wPRS

Reachability HM Property

Instance: (α, β)-(se-,w-)PRS system with the initial state s_0 and an HM formula φ

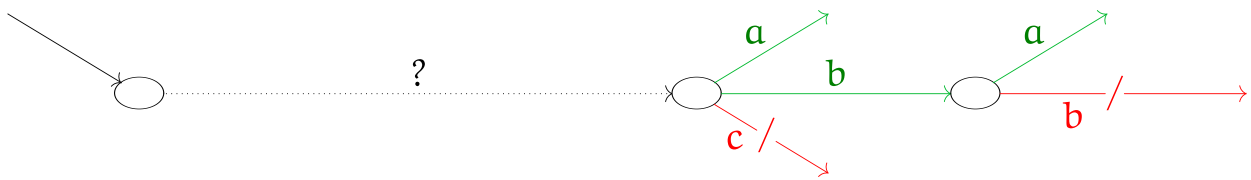
Question: $s_0 \models \mathsf{EF}\varphi$?

HM formula: $\psi = \ tt \ | \ \neg\psi \ | \ \psi_1 \wedge \psi_2 \ | \ \langle a \rangle \psi$

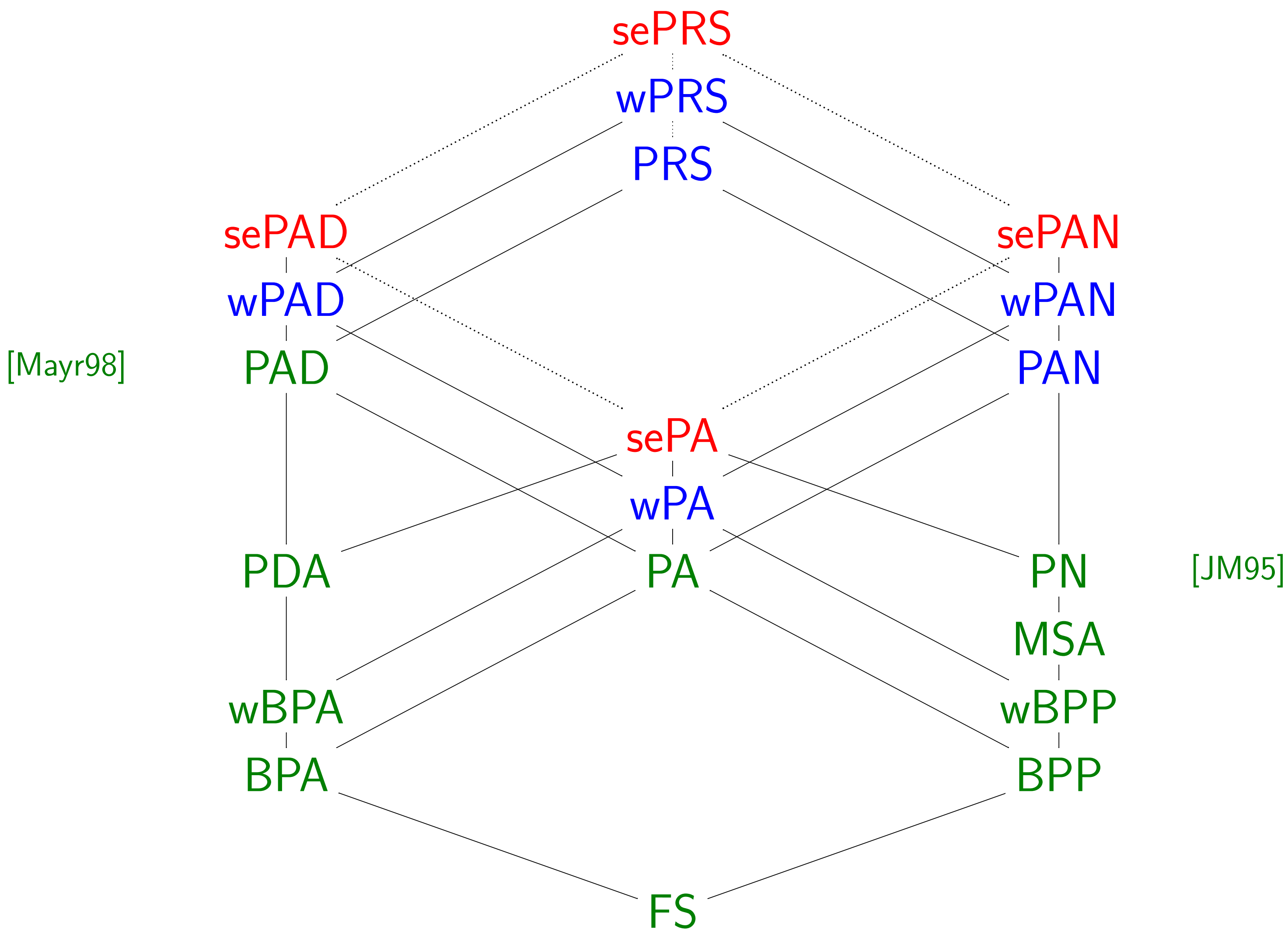
nesting of diamonds

Example:

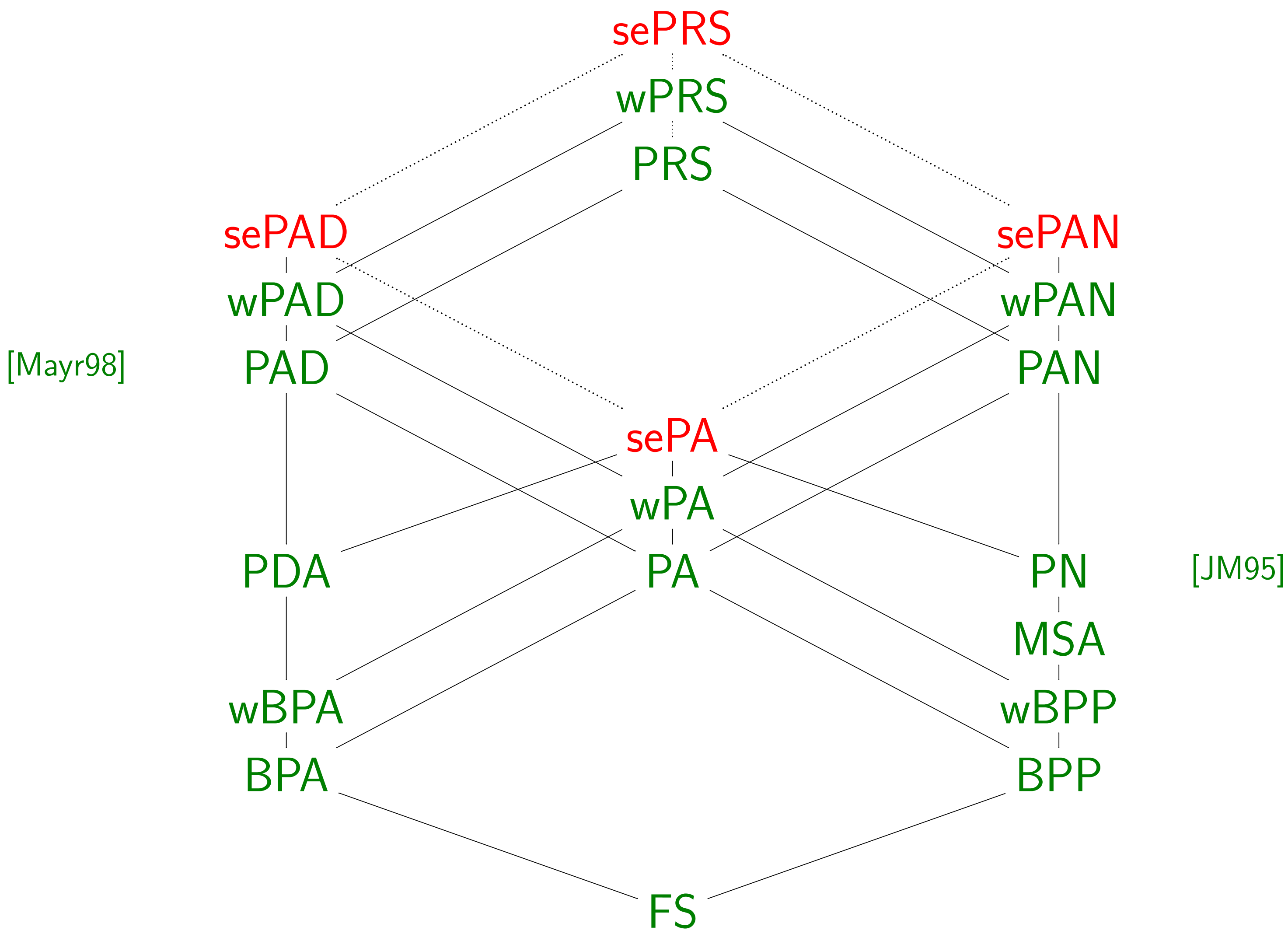
$$s_0 \models \mathsf{EF}(\ \langle a \rangle tt \ \wedge \ \langle b \rangle (\langle a \rangle tt \wedge \neg \langle b \rangle tt) \ \wedge \ \neg \langle c \rangle tt \ )$$



Reachability HM Properties



Reachability HM Properties [FSTTCS'05]



Reachability HM property

Theorem: **Reachability HM property is decidable for wPRS.**

Theorem [JKM01]: Decidability of reachability HM property $\Longrightarrow$ decidability of strong bisimilarity with FS.

Corollary: **Strong bisimilarity between wPRS and FS is decidable.**

(an open question for PAN and PRS since 1998)

Decidability of EF logic

Instance: (α, β)-(se-,w-)PRS system with the initial state s_0 and an EF formula φ

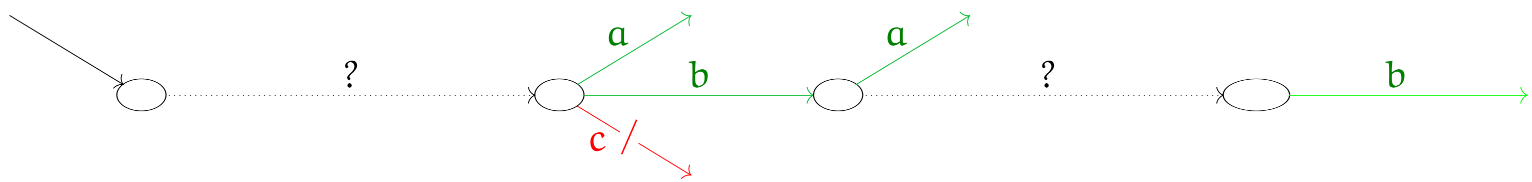
Question: $s_0 \models \varphi$?

HM formula: $\psi = \mathit{tt} \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \langle a \rangle \psi \mid \mathrm{EF}\psi$

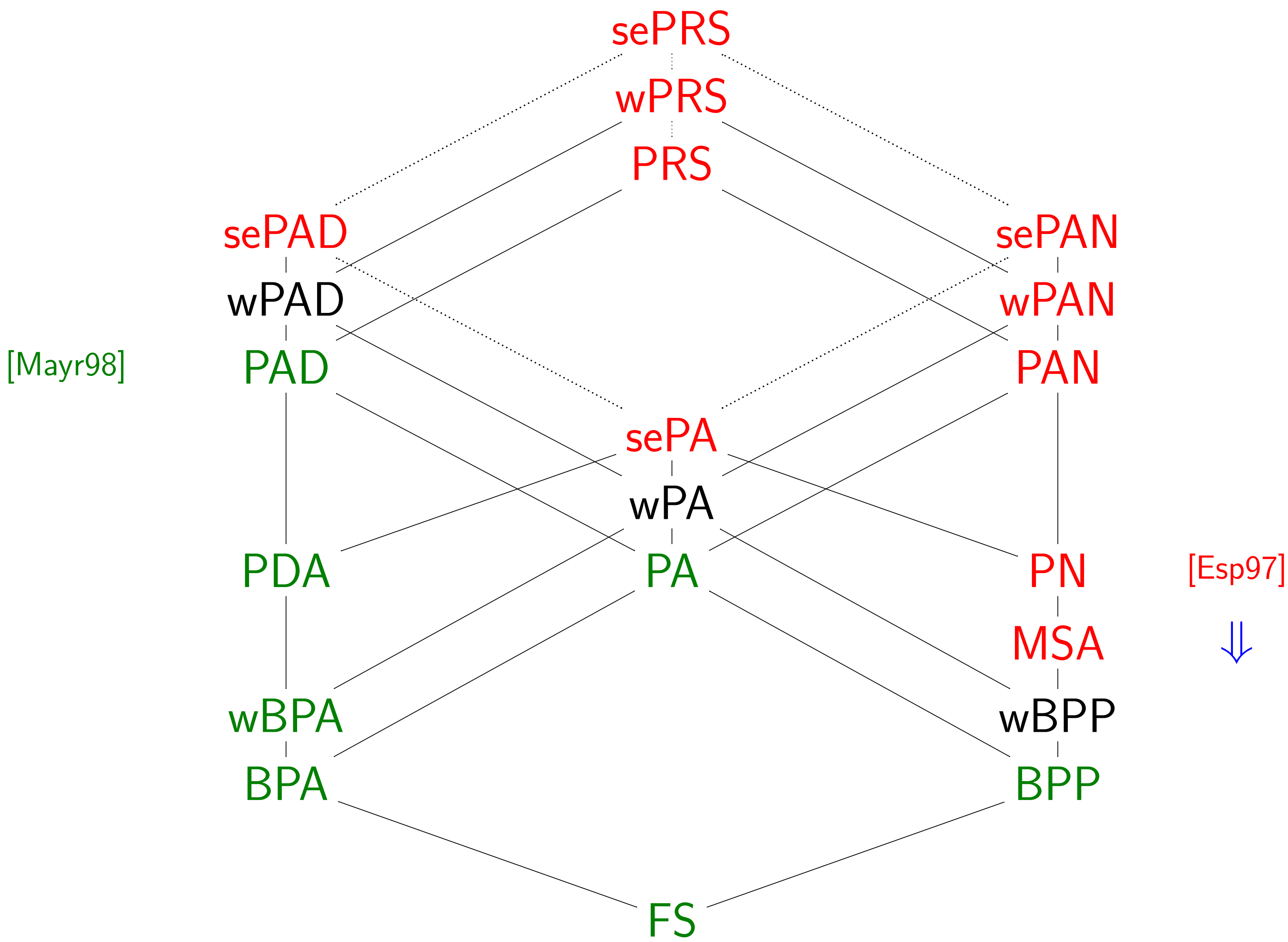
nesting of EF operators

Example:

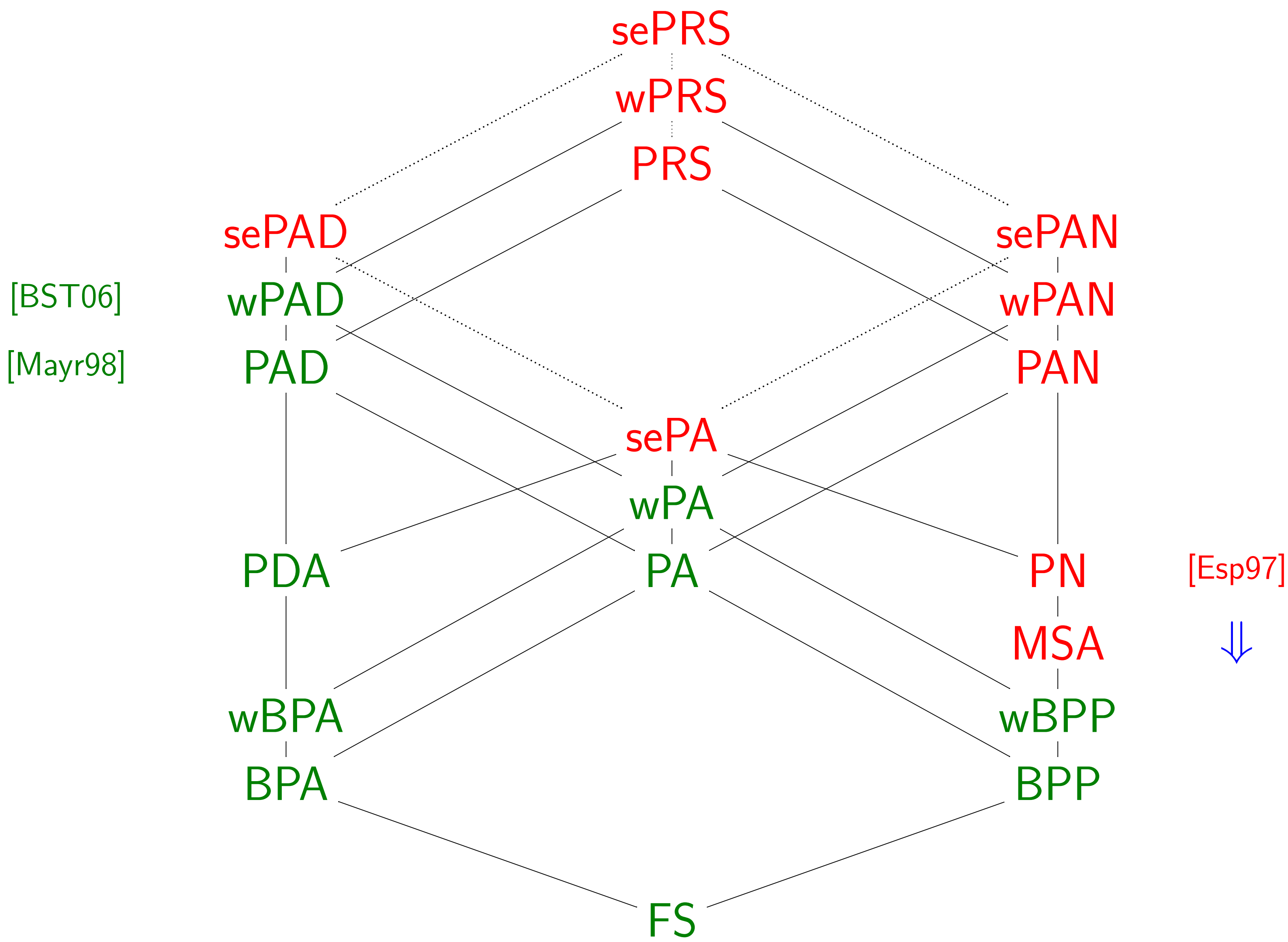
$$s_0 \models \mathrm{EF}(\ \langle a \rangle \mathit{tt} \ \wedge\ \langle b \rangle (\langle a \rangle \mathit{tt} \wedge \mathrm{EF}\langle b \rangle \mathit{tt})\ \wedge\ \neg\langle c \rangle \mathit{tt}\ )$$



Decidability of EF Logic



Decidability of EF Logic



Decidability of Linear Time Logic (LTL)

Instance: (α, β)-(se-,w-)PRS system with the initial state s_0
and an LTL formula φ

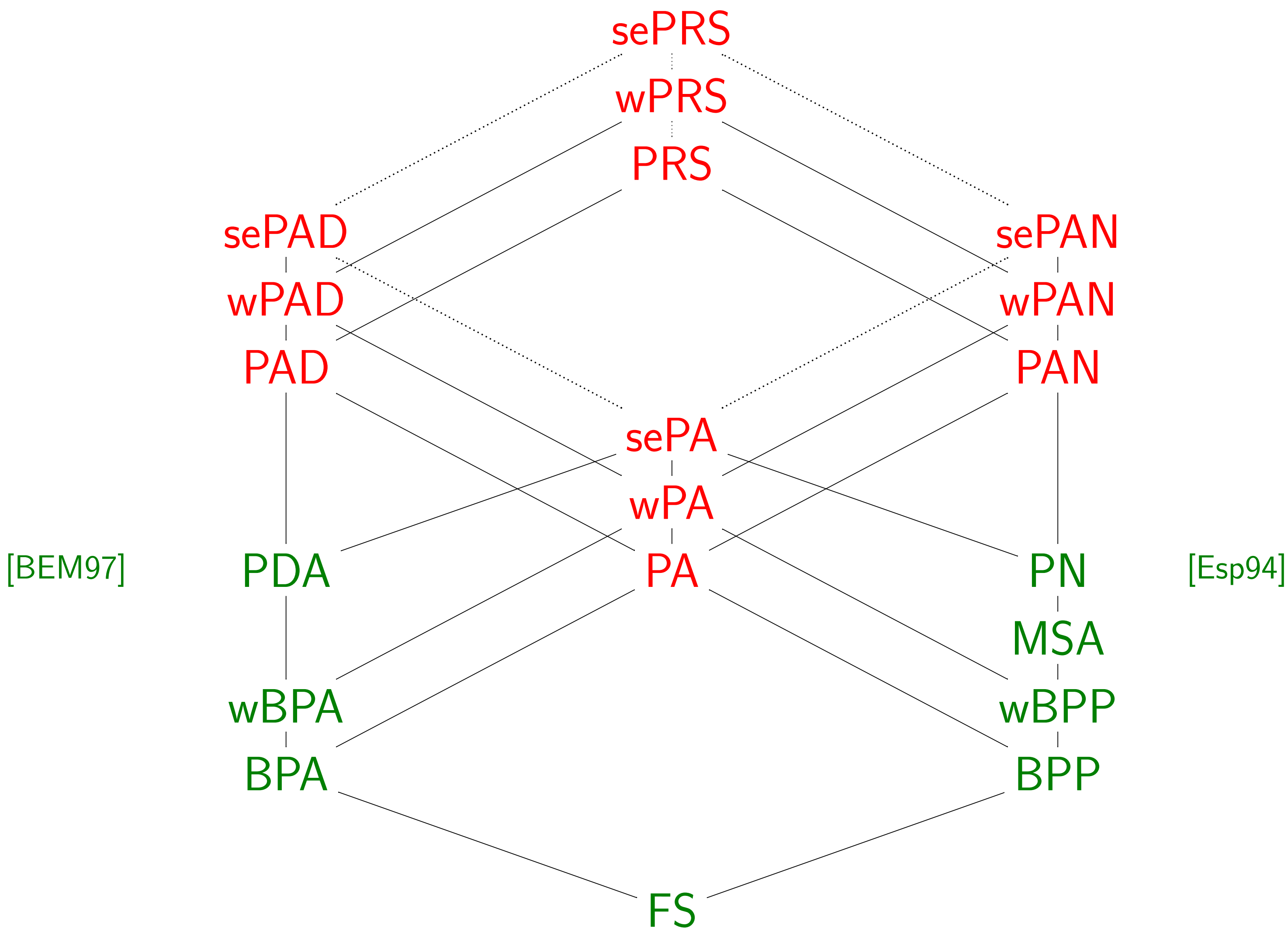
Question: $s_0 \models \varphi$?

LTL formula: $\psi = \; tt \;|\; a \;|\; \neg\psi \;|\; \psi_1 \wedge \psi_2 \;|\; \mathsf{X}\psi \;|\; \psi \,\mathsf{U}\, \psi$
where a is an action

Example:

$\mathsf{X}a$	next	$babacdabdca\ldots$
$a\,\mathsf{U}\,b$	until	$a\ldots abacdab\ldots$

Decidability of LTL



Lamport Logic Definition

Linear Temporal Logic

$$\varphi \ ::= \ tt \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \varphi\, \mathsf{U}\, \varphi.$$

$\mathsf{X}a$	next	$babacdabdca\ldots$
$a\,\mathsf{U}\,b$	until	$a\ldots abacdab\ldots$

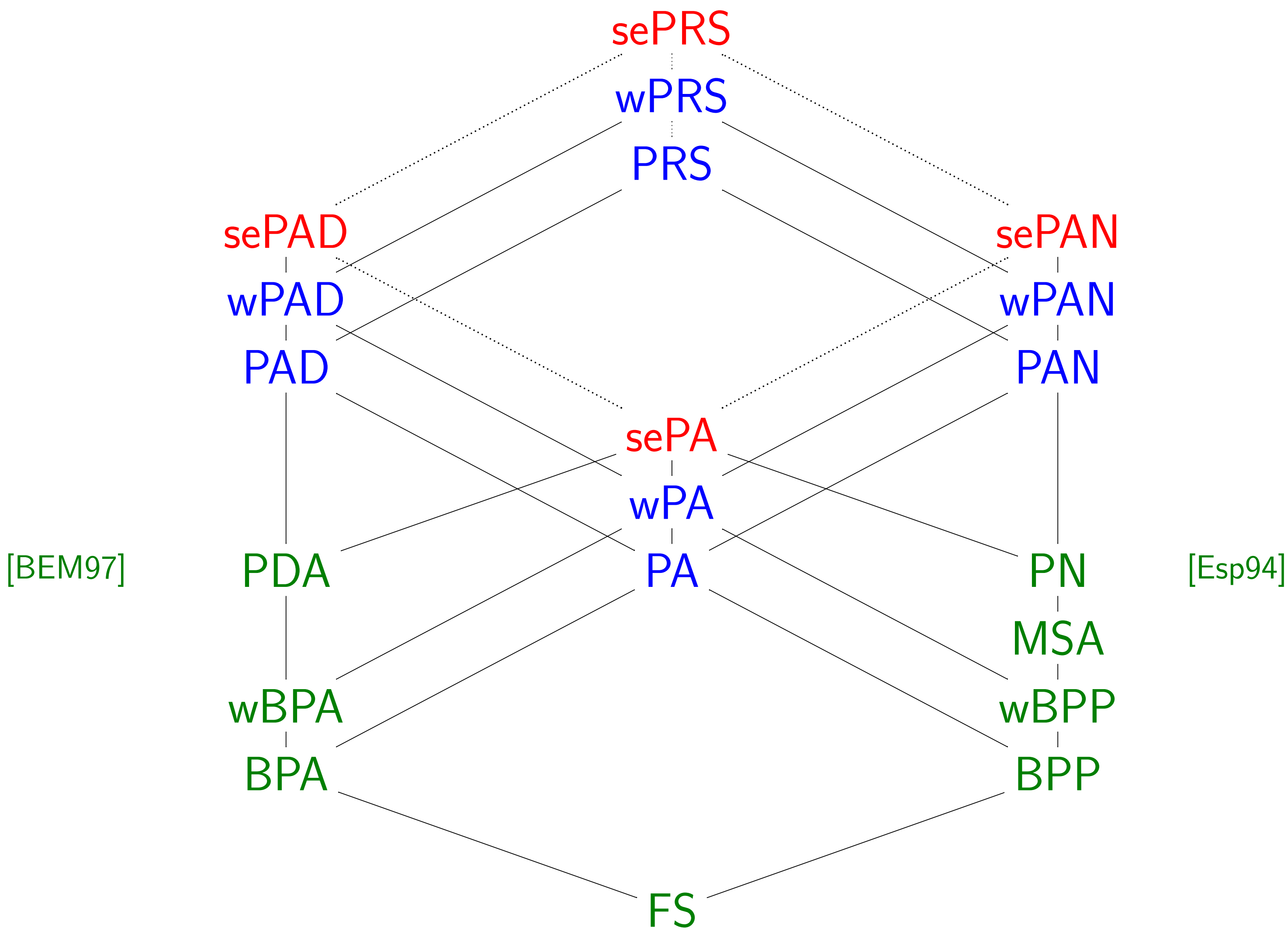
Lamport Logic

$$\varphi \ ::= \ tt \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{F}\varphi.$$

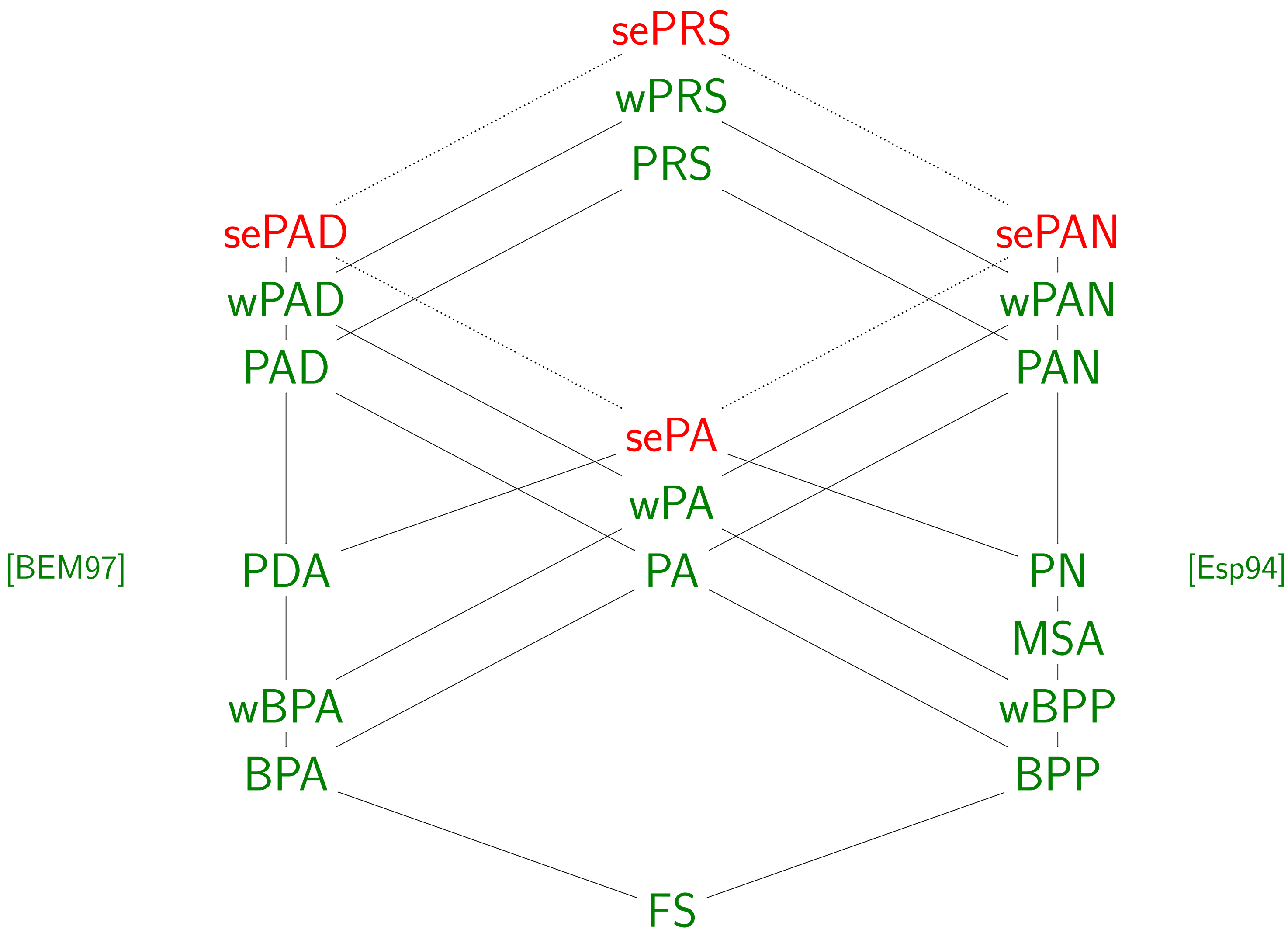
$\mathsf{F}\varphi$	"eventually φ"	$tt\,\mathsf{U}\,\varphi$
$\mathsf{G}\varphi$	"always φ"	$\neg\mathsf{F}\neg\varphi$

liveness: $\mathsf{F}\varphi$ "φ eventually happens", safety: $\mathsf{G}\neg\varphi$ "φ never happens"

Decidability of Lamport Logic



Decidability of Lamport Logic [FSTTCS'06]



Summary

Summary

- introducing more expressive wPRS classes
- reachability problem is decidable for wPRS
- reachability HM property is decidable for wPRS
  bisimulation with FS is decidable for wPRS
- Lamport logic is decidable for wPRS

Corresponding papers

[INFINITY'03] - M. Křetínský, V. Řehák, and J. Strejček: **On Extensions of Process Rewrite Systems: Rewrite Systems with Weak Finite-State Unit**, in INFINITY 2003, ENTCS 98, pp. 75–88. Elsevier, 2004.

[CONCUR'04] - M. Křetínský, V. Řehák, and J. Strejček: **Extended Process Rewrite Systems: Expressiveness and Reachability**, in CONCUR 2004, LNCS 3170, pp. 355–370. Springer, 2004.

[INFINITY'05] - M. Křetínský, V. Řehák, and J. Strejček: **Refining the Undecidability Border of Weak Bisimilarity**, in INFINITY 2005, ENTCS 149:1, pp.17-36, Elsevier, 2006.

[FSTTCS'05] - M. Křetínský, V. Řehák, and J. Strejček: **Reachability of Hennessy-Milner Properties for Weakly Extended PRS**, in FSTTCS 2005, LNCS 3821, pp. 213–224. Springer, 2005.

[FSTTCS'06] - L. Bozzelli, M. Křetínský, V. Řehák, and J. Strejček: **On Decidability of LTL Model Checking for Process Rewrite Systems**, in FSTTCS 2006, LNCS 4337, pp. 248–259. Springer, 2006.