Compositional analysis of modular systems using hierarchical state space abstraction

Yves-Stan Le Cornec

advisor : Franck Pommereau

MeFoSyLoMa
IBISC, University of Evry

19 april 2013

Motivation

- Model checking
  - μ-calculus formula
  - modular Petri net
    - fused transitions
    - finite semantics of the modules
- Exploit modularity to alleviate the state space explosion problem
  - incremental approach
  - try to conclude as soon as possible
  - formula-dependent hierarchical reductions

Modal μ-calculus

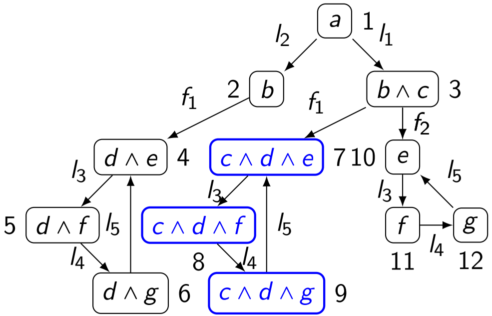
subsumes CTL, LTL, CTL*, ...

Syntax: $\varphi \quad ::= \quad B \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\alpha\rangle\varphi \mid \mu X.\varphi \mid X$

Modal μ-calculus

subsumes CTL, LTL, CTL*, ...

Syntax: φ ::= $\mathbf{B} \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\alpha\rangle\varphi \mid \mu X.\varphi \mid X$

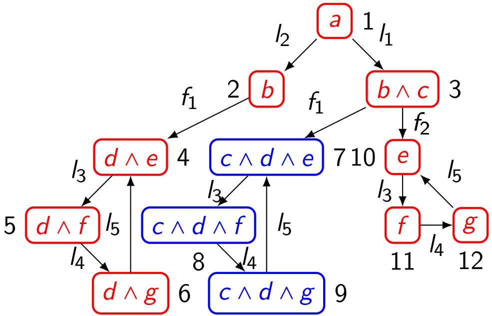


$\varphi = B = c \wedge d$

Modal μ-calculus

subsumes CTL, LTL, CTL*, ...

Syntax: $\varphi \quad ::= \quad B \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\alpha\rangle\varphi \mid \mu X.\varphi \mid X$

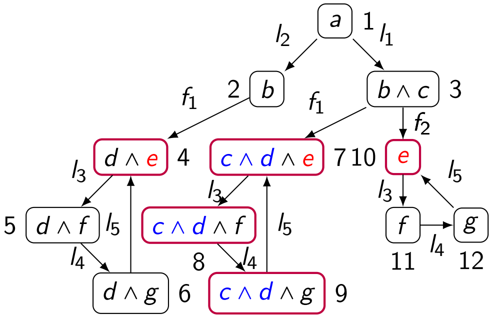


$\varphi = \neg(c \wedge d)$

Modal μ-calculus

subsumes CTL, LTL, CTL*, ...

Syntax: $\varphi \quad ::= \quad B \mid \neg\varphi \mid \boldsymbol{\varphi_1} \vee \boldsymbol{\varphi_2} \mid \langle\alpha\rangle\varphi \mid \mu X.\varphi \mid X$

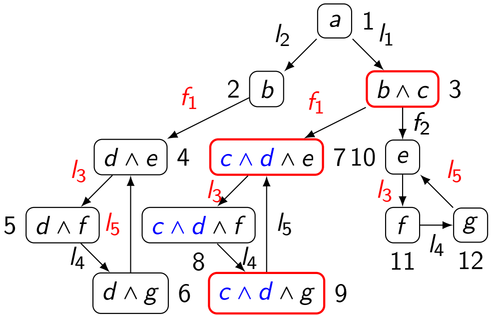


$\varphi = (c \wedge d) \vee (e)$

Modal μ-calculus

subsumes CTL, LTL, CTL*, ...

Syntax: $\varphi \quad ::= \quad B \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\alpha\rangle\varphi \mid \mu X.\varphi \mid X$

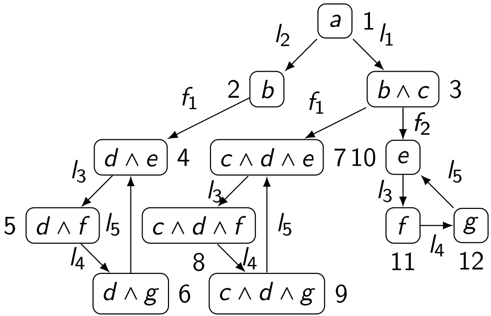


$\varphi = \langle f_1, l_3, l_5\rangle(c \wedge d)$

Modal μ-calculus

subsumes CTL, LTL, CTL*, . . .

Syntax: $\varphi ::= B \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\alpha\rangle\varphi \mid \boldsymbol{\mu X.\varphi} \mid \mathbf{X}$



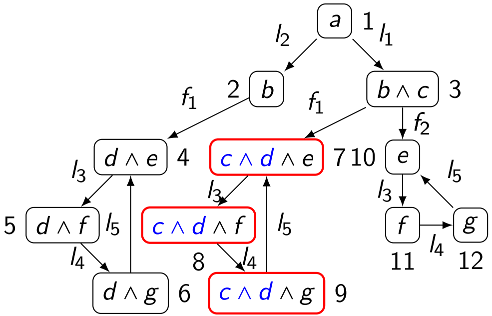
$\varphi = \mu X.\psi$ where

$\psi = ((c \wedge d) \vee \langle\rangle X)$

Modal μ-calculus

subsumes CTL, LTL, CTL*, ...

Syntax: $\varphi \quad ::= \quad B \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\alpha\rangle\varphi \mid \boldsymbol{\mu X.\varphi} \mid \mathbf{X}$



$\varphi = \mu X.\psi$ where

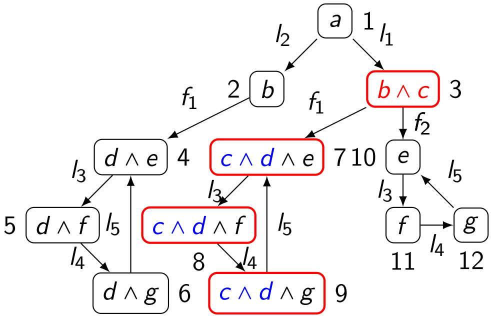
$\psi = ((c \wedge d) \vee \langle\rangle X)$

$\psi(\varnothing)$

Modal μ-calculus

subsumes CTL, LTL, CTL*, . . .

Syntax: $\varphi \quad ::= \quad B \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\alpha\rangle\varphi \mid \boldsymbol{\mu X.\varphi} \mid \mathbf{X}$



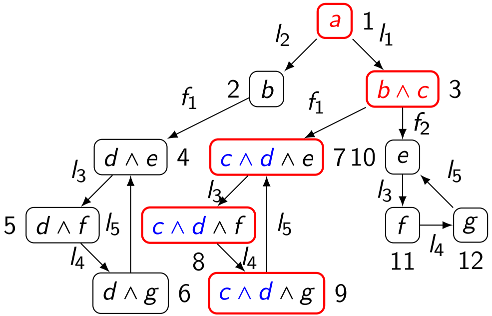
$\psi(\psi(\varnothing))$

$\varphi = \mu X.\psi$ where
$\psi = ((c \wedge d) \vee \langle\rangle X)$

Modal μ-calculus

subsumes CTL, LTL, CTL*, . . .

Syntax: $\varphi \quad ::= \quad B \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\alpha\rangle\varphi \mid \boldsymbol{\mu X.\varphi} \mid \mathbf{X}$



$\psi(\psi(\psi(\varnothing)))$

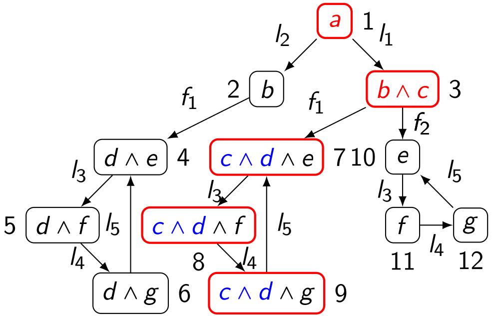
$\varphi = \mu X.\psi$ where

$\psi = ((c \wedge d) \vee \langle\rangle X)$

Modal μ-calculus

subsumes CTL, LTL, CTL*, . . .

Syntax: $\varphi ::= B \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\alpha\rangle\varphi \mid \boldsymbol{\mu X.\varphi} \mid \mathbf{X}$



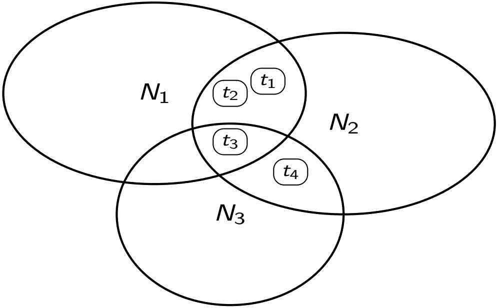
$\varphi = \mu X.\psi$ where

$\psi = ((c \wedge d) \vee \langle\rangle X)$

$\psi(\psi(\psi(\psi(\varnothing))))$

Modular Petri nets

Global system:
collection of Petri nets with fused transitions.

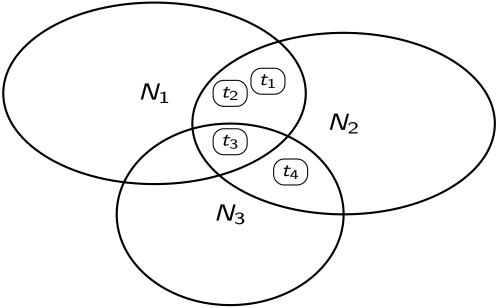


Theorem (Semantics consistency) [Christensen and Petrucci, 1992]

$$[\![N_1 \oplus \cdots \oplus N_n]\!] \quad \simeq \quad [\![N_1]\!] \otimes \cdots \otimes [\![N_n]\!]$$

Modular Petri nets

Global system:
collection of Petri nets with fused transitions.



Theorem (Semantics consistency) [Christensen and Petrucci, 1992]

$$[\![N_1 \oplus \cdots \oplus N_n]\!] \quad \simeq \quad [\![N_1]\!] \otimes \cdots \otimes [\![N_n]\!]$$

Application: semantics of net $(N_1, \ldots, N_5)$ computed equivalently as :

- $[\![N_1 \oplus \cdots \oplus N_5]\!]$
- $[\![N_1 \oplus N_2]\!] \otimes [\![N_3 \oplus N_4]\!] \otimes [\![N_5]\!]$
- $\cdots$

Exploiting the modularity

- Local verification of the formula : $\varphi \overset{?}{\dashv} S_i$

Exploiting the modularity

- Local verification of the formula : $\varphi \overset{?}{\dashv} S_i$
- Formula dependent abstraction
  - A reduction operation : $\lfloor S \rfloor_{\varphi}$
  - An equivalence relation on the transition systems: $\sim^{\varphi}$

Exploiting the modularity

- Local verification of the formula : $\varphi \overset{?}{\dashv} S_i$
- Formula dependent abstraction
  - A reduction operation : $\lfloor S \rfloor_\varphi$
  - An equivalence relation on the transition systems: $\sim^\varphi$

Theorem ($\sim^\varphi$ preserves the truth value of φ)

1. $\lfloor S \rfloor_\varphi \sim^\varphi S$
2. *If* $S_i \sim^\varphi S'_i$ *for all* i *then* $S_1 \otimes \cdots \otimes S_n \sim^\varphi S'_1 \otimes \cdots \otimes S'_n$
3. *If* $S_1 \sim^\varphi S_2$ *then* $\varphi \overset{?}{\dashv} S_1$ *iff* $\varphi \overset{?}{\dashv} S_2$ *and* $S_1 \models \varphi$ *iff* $S_2 \models \varphi$

Exploiting the modularity

- Local verification of the formula : $\varphi \overset{?}{\dashv} S_i$
- Formula dependent abstraction
  - A reduction operation : $\lfloor S \rfloor_\varphi$
  - An equivalence relation on the transition systems: $\sim^\varphi$

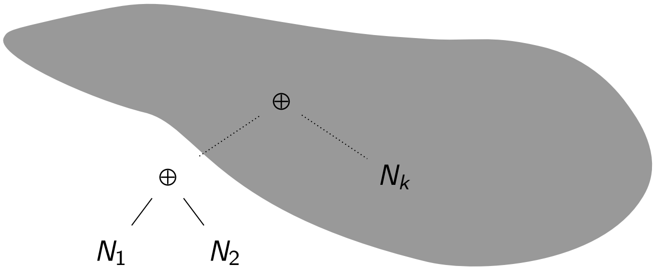
Theorem ($\sim^\varphi$ preserves the truth value of φ)

1. $\lfloor S \rfloor_\varphi \sim^\varphi S$
2. *If* $S_i \sim^\varphi S'_i$ *for all* i *then* $S_1 \otimes \cdots \otimes S_n \sim^\varphi S'_1 \otimes \cdots \otimes S'_n$
3. *If* $S_1 \sim^\varphi S_2$ *then* $\varphi \overset{?}{\dashv} S_1$ *iff* $\varphi \overset{?}{\dashv} S_2$ *and* $S_1 \models \varphi$ *iff* $S_2 \models \varphi$

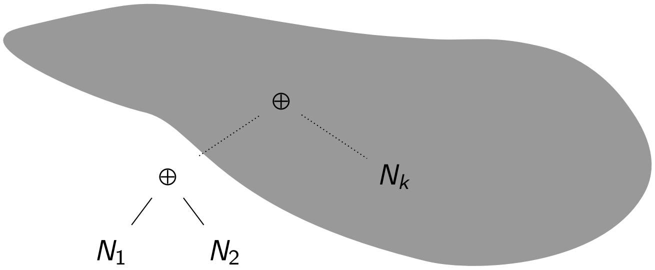
Application: with $\lfloor\!\lfloor N_n \rfloor\!\rfloor_\varphi \overset{\mathrm{df}}{=} \lfloor [\![N_n]\!] \rfloor_\varphi$ we have:

$$[\![N_1 \oplus \cdots \oplus N_n]\!] \quad \simeq \quad [\![N_1]\!] \otimes \cdots \otimes [\![N_n]\!] \quad \sim^\varphi \quad \lfloor\!\lfloor N_1 \rfloor\!\rfloor_\varphi \otimes \cdots \otimes \lfloor\!\lfloor N_n \rfloor\!\rfloor_\varphi$$

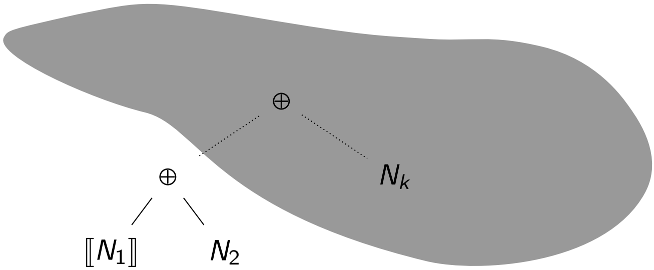
Example



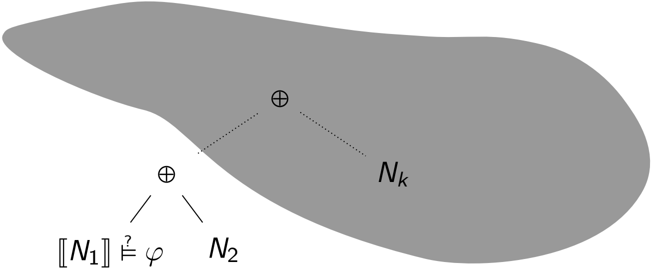
Example



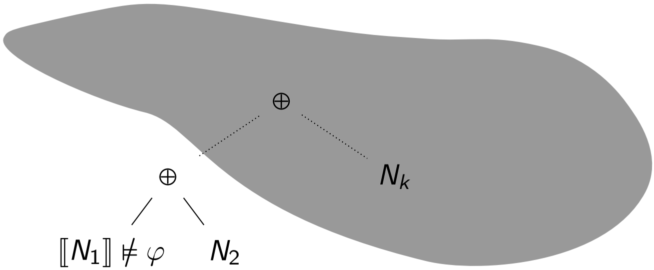
Example



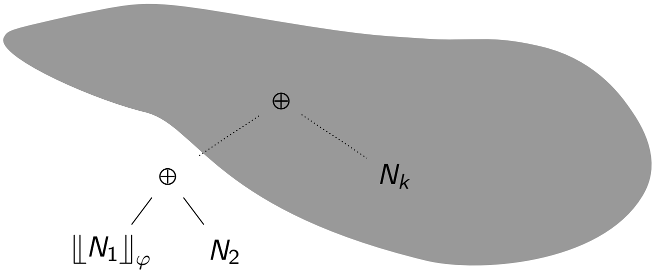
Example



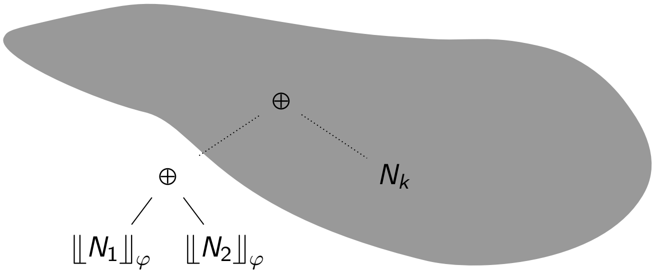
Example



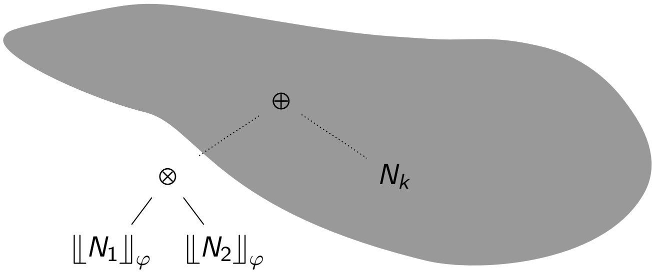
Example



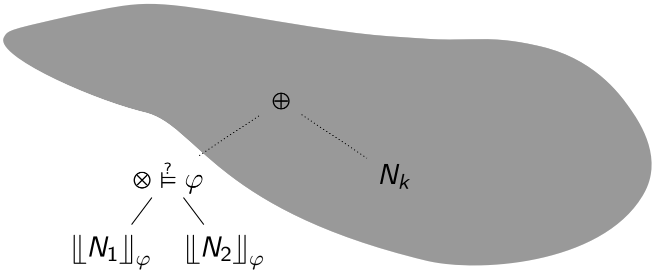
Example



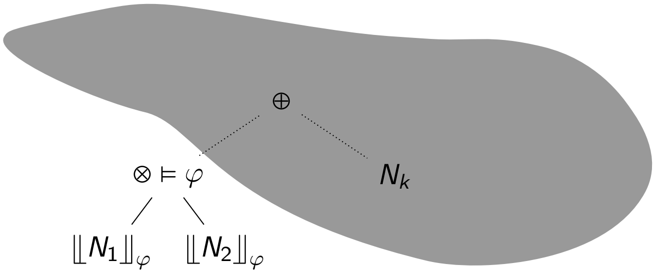
Example



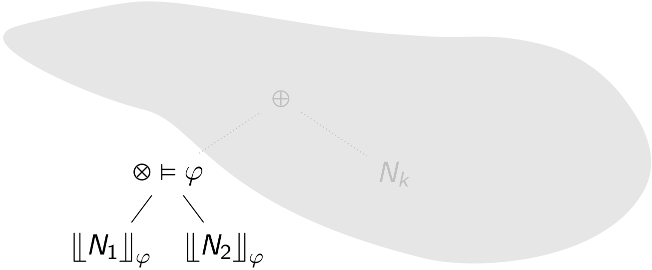
Example



Example



Example



Construction of $\llbracket\ \rrbracket_\varphi$

- Goal : $\forall env, ((x, env) \vDash \varphi \Leftrightarrow (y, env) \vDash \varphi)$

Construction of $\llbracket\ \rrbracket_\varphi$

- Goal : $\forall env, ((x, env) \vDash \varphi \Leftrightarrow (y, env) \vDash \varphi)$
- Goal : $\forall env, ((x, env) \vDash \varphi \Rightarrow (y, env) \vDash \varphi)$

Construction of $\llbracket\ \rrbracket_{\varphi}$

- Goal : $\forall env, ((x, env) \vDash \varphi \Leftrightarrow (y, env) \vDash \varphi)$
- Goal : $\forall env, ((x, env) \vDash \varphi \Rightarrow (y, env) \vDash \varphi)$
- Goal : $\forall env, ((x, env) \vDash \psi \Rightarrow (y, env) \vDash \varphi)$

Construction of $\llbracket \ \rrbracket_\varphi$

- Goal : $\forall env, ((x, env) \vDash \varphi \Leftrightarrow (y, env) \vDash \varphi)$
- Goal : $\forall env, ((x, env) \vDash \varphi \Rightarrow (y, env) \vDash \varphi)$
- Goal : $\forall env, ((x, env) \vDash \psi \Rightarrow (y, env) \vDash \varphi)$
- Goal : If $env \vDash$? Then $(x, env) \vDash \psi \Rightarrow (y, env) \vDash \varphi$

Construction of $[\![\ ]\!]_\varphi$

- Goal : $\forall env, ((x, env) \vDash \varphi \Leftrightarrow (y, env) \vDash \varphi)$
- Goal : $\forall env, ((x, env) \vDash \varphi \Rightarrow (y, env) \vDash \varphi)$
- Goal : $\forall env, ((x, env) \vDash \psi \Rightarrow (y, env) \vDash \varphi)$
- Goal : If $env \vDash$? Then $(x, env) \vDash \psi \Rightarrow (y, env) \vDash \varphi$
- Goal : If $env \vDash \langle\!\langle \psi, \varphi \rangle\!\rangle(x, y)$ Then $(x, env) \vDash \psi \Rightarrow (y, env) \vDash \varphi$

- We build $\langle\!\langle \psi, \varphi \rangle\!\rangle_S : Q^2 \rightarrow \mathbb{B}(Ex)$ where
- Q states of module S
- Ex state variables in ψ and φ external to S

Inductive construction of $\langle\!\langle \psi, \varphi \rangle\!\rangle$

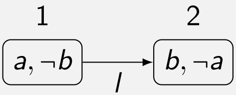
Base case

$$\langle\!\langle B_1, B_2 \rangle\!\rangle(x, y) \stackrel{\text{df}}{=} B_1@x \Rightarrow B_2@y = \neg B_1@x \vee B_2@y$$

Example: $\langle\!\langle B, B \rangle\!\rangle$

- where $B = (a \vee v) \wedge (b \vee w)$
- a and b are local variables
- v and w are external variables

LTS



we have $B@1 = w$ and $B@2 = v$ so :

$$\langle\!\langle B, B \rangle\!\rangle = \begin{cases} (1,1);(2,2) & \mapsto \top \\ (1,2) & \mapsto \neg w \vee v \\ (2,1) & \mapsto \neg v \vee w \end{cases}$$

Other rules

Disjunction rule

$$\langle\!\langle \psi, \varphi_1 \vee \varphi_2 \rangle\!\rangle(x, y) = \langle\!\langle \psi, \varphi_1 \rangle\!\rangle(x, y) \vee \langle\!\langle \psi, \varphi_2 \rangle\!\rangle(x, y)$$

Local next rule

$$\langle\!\langle \psi, \langle l \rangle \varphi \rangle\!\rangle(x, y) \stackrel{\text{df}}{=} \left(\bigvee_{y \xrightarrow{l} y'} \langle\!\langle \psi, \varphi \rangle\!\rangle(x, y')\right) \vee \langle\!\langle \psi, \bot \rangle\!\rangle(x, y)$$

Synchronized next rule

$$\langle\!\langle \langle s \rangle \psi, \langle s \rangle \varphi \rangle\!\rangle(x, y) = \begin{cases} \top & \text{iff } \bigwedge_{x \xrightarrow{s} x'} \bigvee_{y \xrightarrow{s} y'} \langle\!\langle \psi, \varphi \rangle\!\rangle(x', y') = \top \\ \bot & \text{otherwise} \end{cases}$$

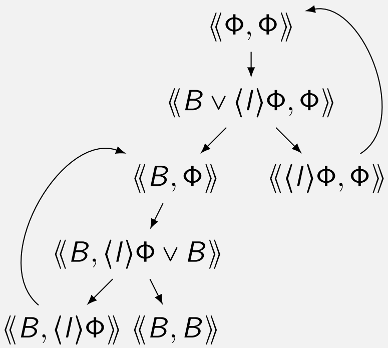
Fixed point rule

$$\langle\!\langle \mu X \psi, \varphi \rangle\!\rangle = \langle\!\langle \psi(\mu X \psi), \varphi \rangle\!\rangle$$

Dependency graph : what needs to be computed

- $\Phi = \mu X . \langle l \rangle X \vee B$
- l is a local action
- a and b are local variables
- v and w are external variables

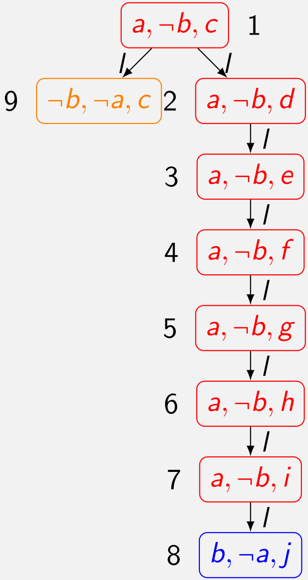
dependency graph of $\langle\langle \Phi, \Phi \rangle\rangle$



Theorem

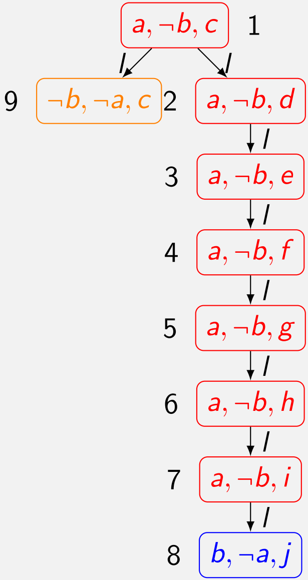
The fixed-point computations converge

Module S



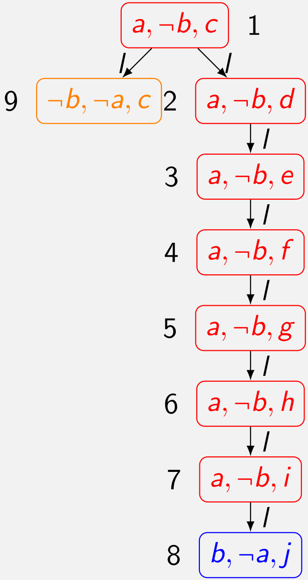
- $\Phi = \mu X.\langle\rangle X \vee B$
- $B = (a \wedge v) \vee (b \wedge w)$

Module S



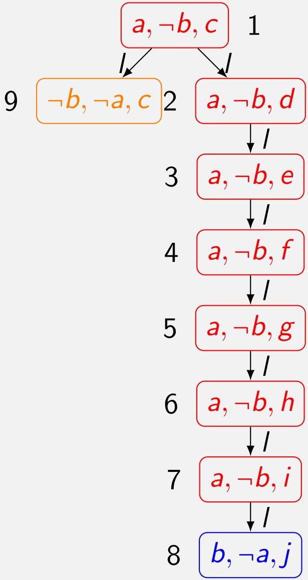
- $\Phi = \mu X.\langle\rangle X \vee B$
- $B = (a \wedge v) \vee (b \wedge w)$
- if $v = \top$ then $1 - 7 \vDash B$ so $1 - 7 \vDash \Phi$

Module S



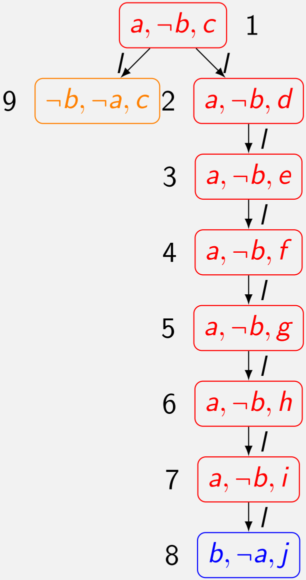
- $\Phi = \mu X.\langle\rangle X \vee B$
- $B = (a \wedge v) \vee (b \wedge w)$
- if $v = \top$ then $1-7 \models B$ so $1-7 \models \Phi$
- if $w = \top$ then $8 \models B$ so $1-7 \models \Phi$

Module S



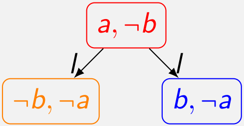
- $\Phi = \mu X.\langle\rangle X \vee B$
- $B = (a \wedge v) \vee (b \wedge w)$
- if $v = \top$ then $1 - 7 \models B$ so $1 - 7 \models \Phi$
- if $w = \top$ then $8 \models B$ so $1 - 7 \models \Phi$
- if $w = \bot$ and $v = \bot$ then $1 - 7 \not\models \Phi$

Module S



- $\Phi = \mu X.\langle\rangle X \vee B$
- $B = (a \wedge v) \vee (b \wedge w)$
- if $v = \top$ then $1 - 7 \models B$ so $1 - 7 \models \Phi$
- if $w = \top$ then $8 \models B$ so $1 - 7 \models \Phi$
- if $w = \bot$ and $v = \bot$ then $1 - 7 \not\models \Phi$

reduced module $\llbracket S \rrbracket_\Phi$



Conclusion

- A framework to incrementally analyze modular system using formula dependent reductions
- Next step:
  good/optimal hierarchical decomposition
  $\Rightarrow$ stop the analysis as soon as possible
- Implementation
- Case studies