POPL 2017 18 January 2017

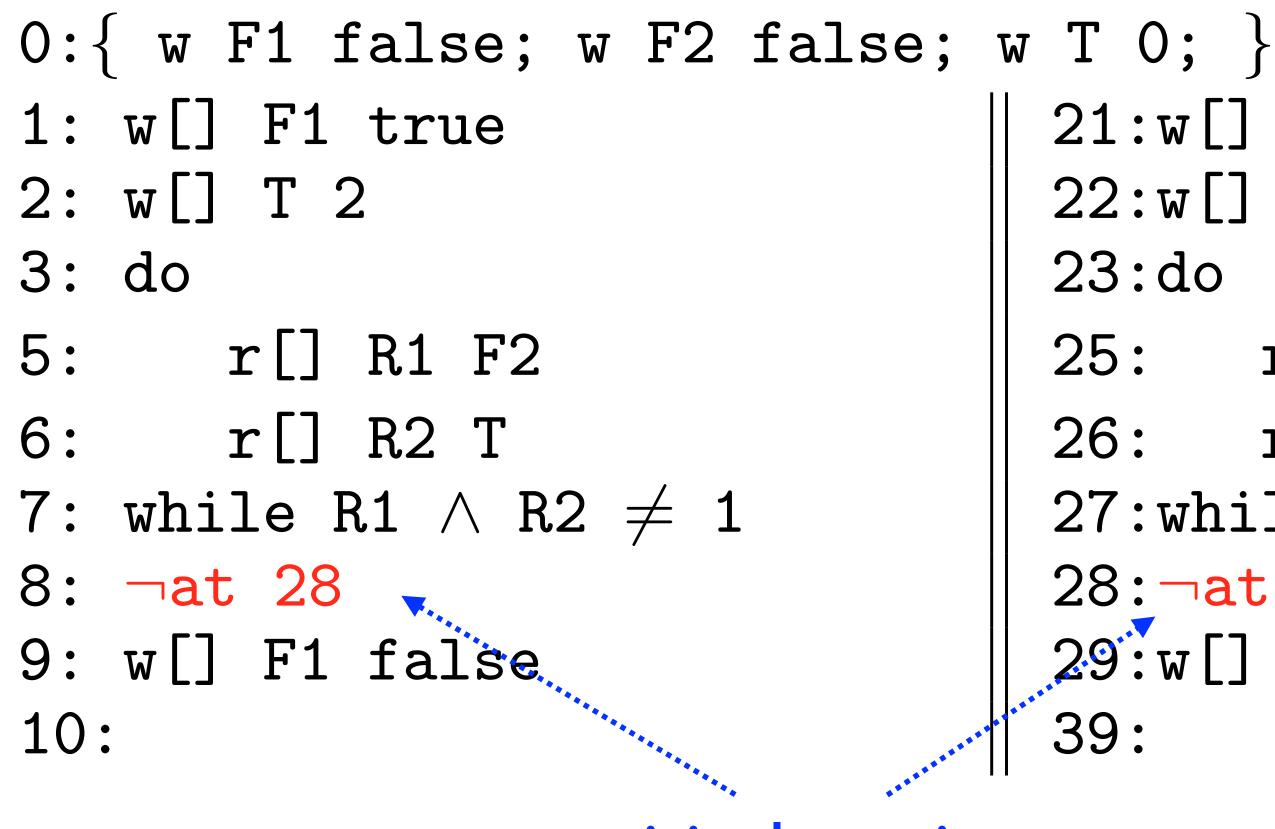
Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017

Ogre et Pythia: An invariance proof method for weak consistency models

ade Alglave (MSR-Cambridge, UCL, UK) Patrick Cousot (NYU, Emer. ENS, PSL)



Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017



critical section

### Example

21:w[] F2 true; 22:w[] T 1; 23:do 25: r[] R3 F1; 26: r[] R4 T; 27:while R3  $\land$  R4  $\neq$  2; 28: ¬at 8 29:w[] F2 false; 39:

### An invariance proof method for WCMs

- Extend Lamport's invariance proof method for parallel programs from sequentially consistent to weak consistency models so that
  - The weak consistency model is a *parameter* of the proof
  - We don't have to redo the whole proof when changing the consistency model

# programs counters

Note: Owicki & Gries is Lamport with auxiliary variables instead of

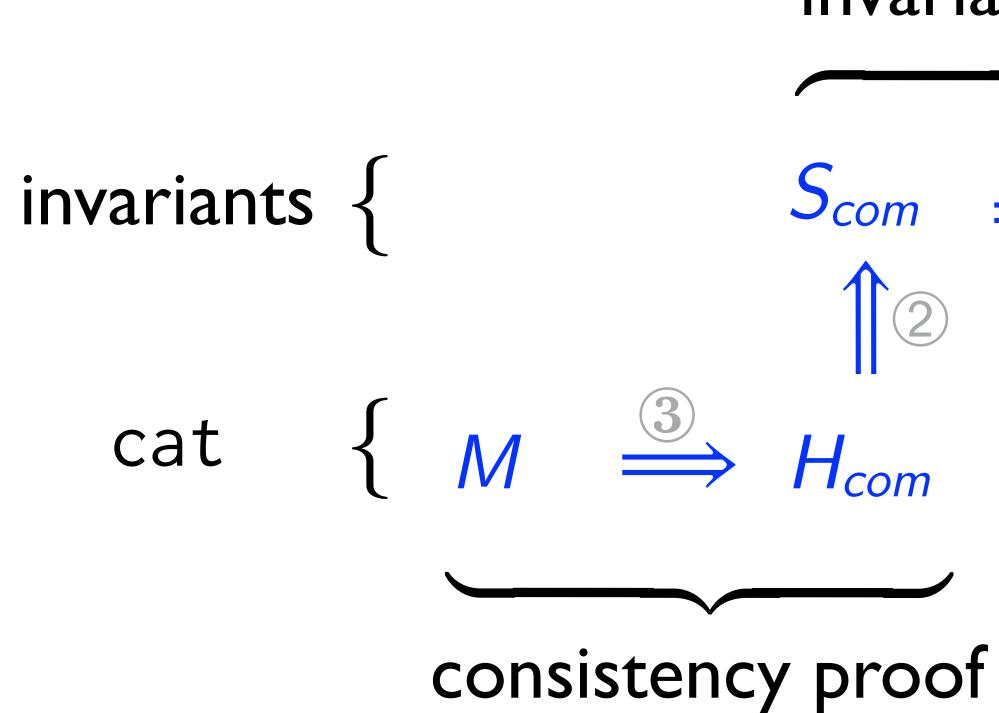


## Separating invariance from WCM

- The invariance proof (that a specification  $S_{inv}$  is invariant for a program):
  - Done for a program consistency hypothesis  $S_{com}$ :
    - Sufficient for the program to be correct
    - Or better, also necessary for correctness (weakest consistency model)
  - This program consistency hypothesis  $S_{com}$  is expressed as an invariant • Sound and (relatively) complete

# Separating invariance from WCM

- Consistency proof:
  - a. The program consistency hypothesis  $S_{com}$  is strengthen into  $H_{com}$  written in a consistency specification language (e.g. cat)
  - b. A cat architecture consistency model M is shown to imply the cat program consistency model  $H_{com}$
- only b. to be redone when changing the architecture
- sound but possibly incomplete



Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017



# invariance proof $\begin{array}{ccc} S_{com} & \stackrel{(1)}{\Longrightarrow} & S_{inv} \end{array}$ 12

Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017

The invariance proof method is designed by abstract interpretation of an analytic semantics

### Analytic semantics

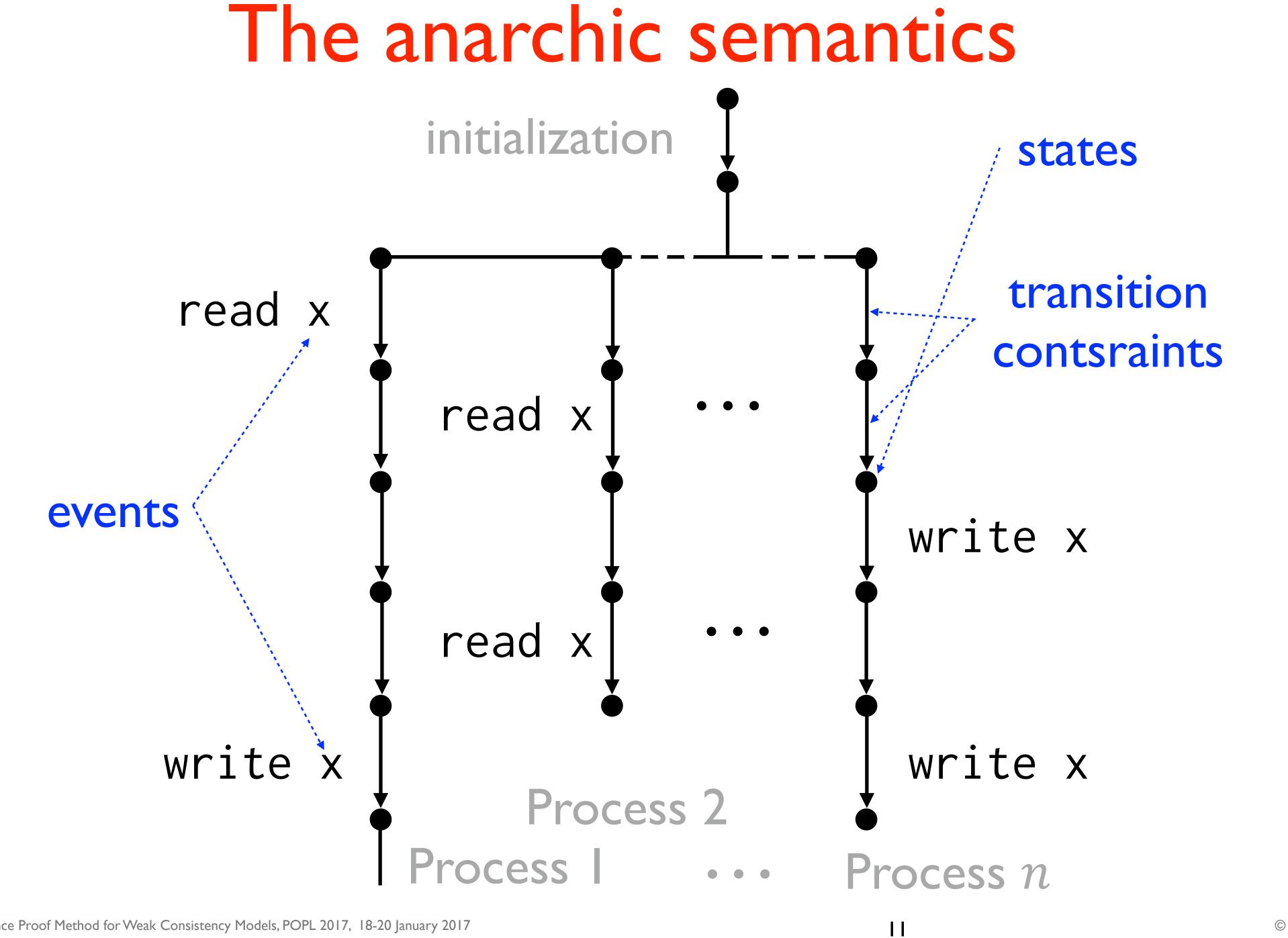
### Anarchic semantics

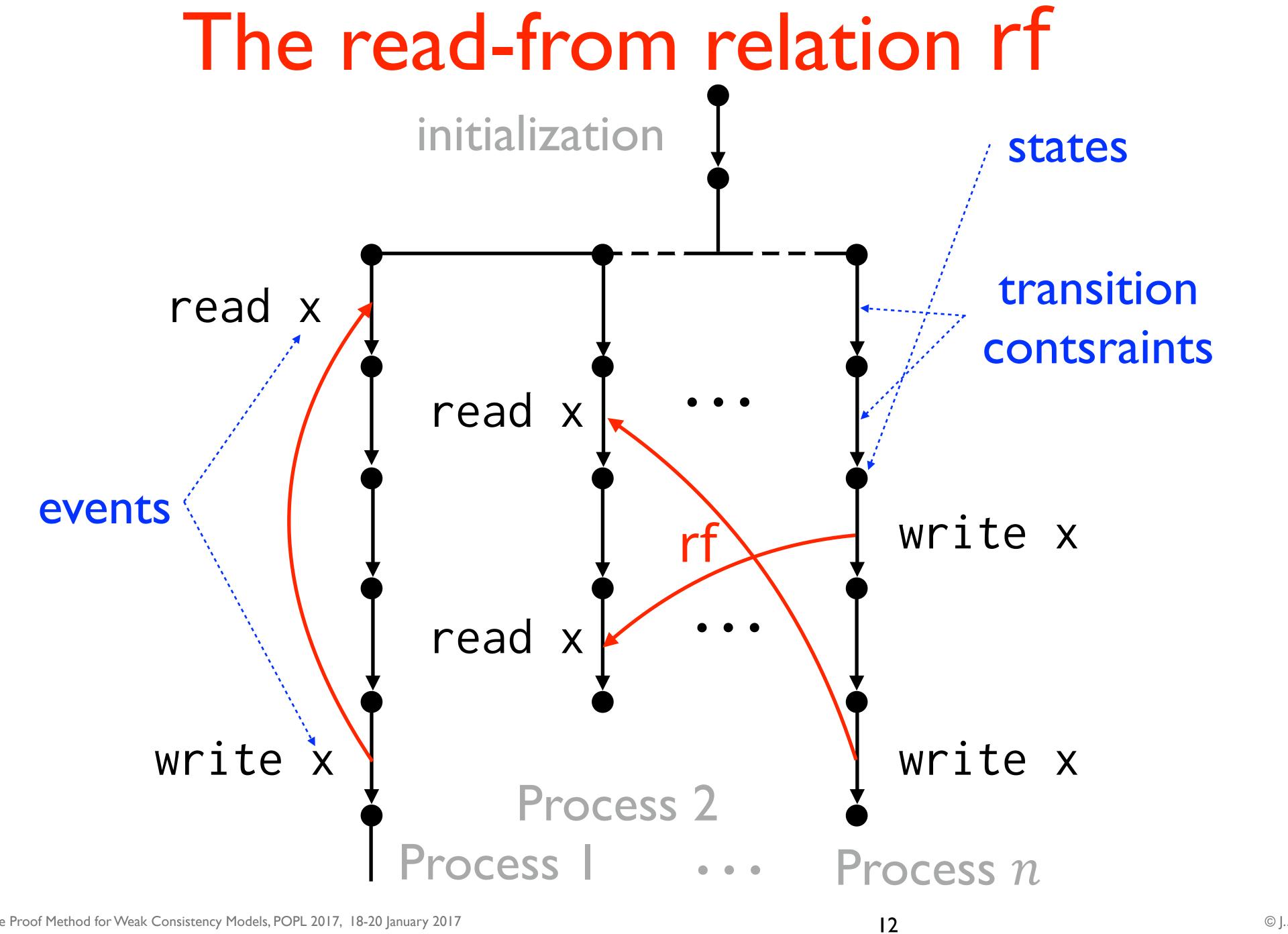
## Weak consistency model

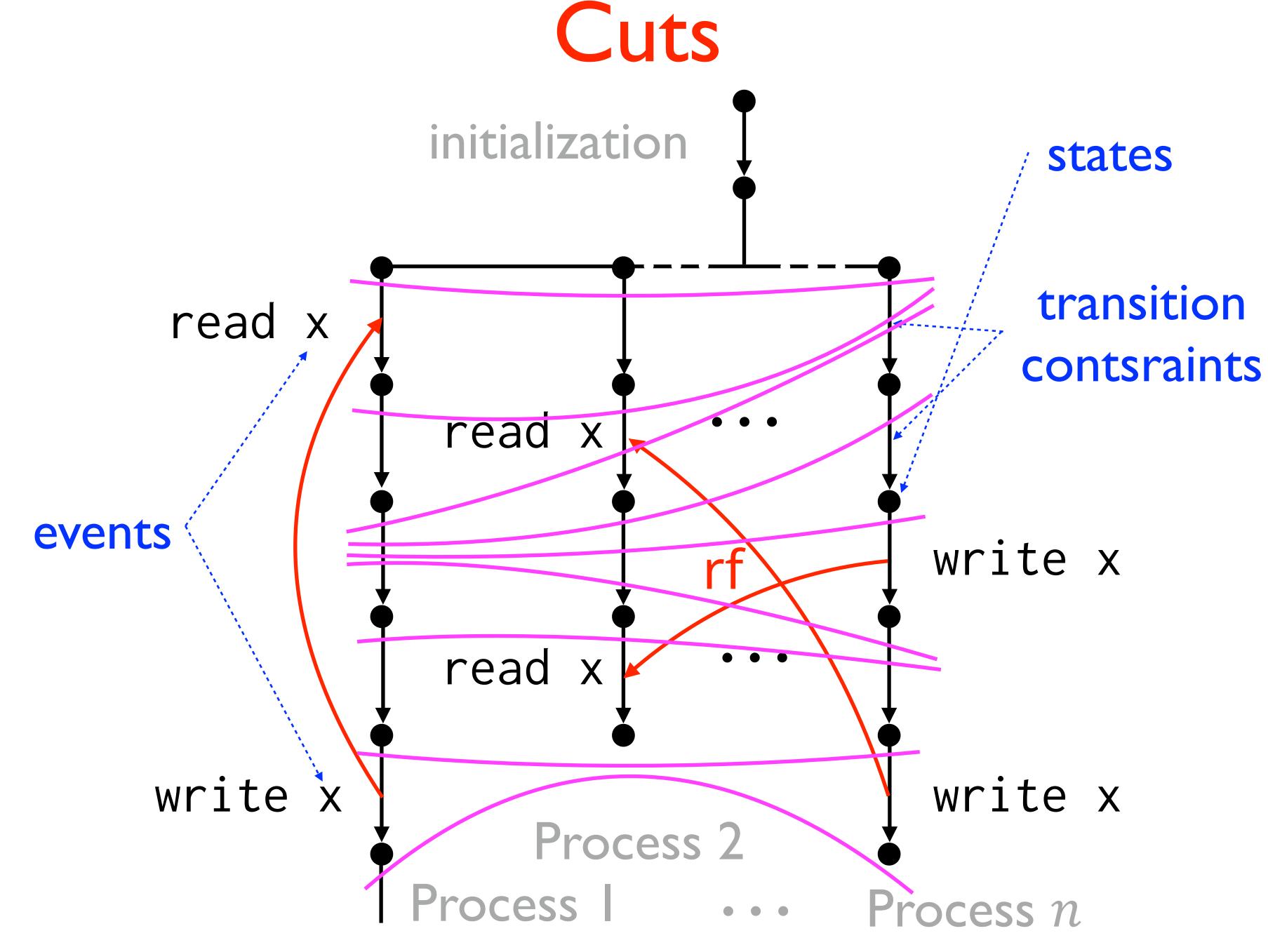
Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017

Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017

The anarchic semantics







### Anarchic semantics of fences

- The anarchic semantics of (localized) fences is skip (the state is unmodified)
- read-from relation rf

• Fences are static marker events used by the WCM in cat to restrict the

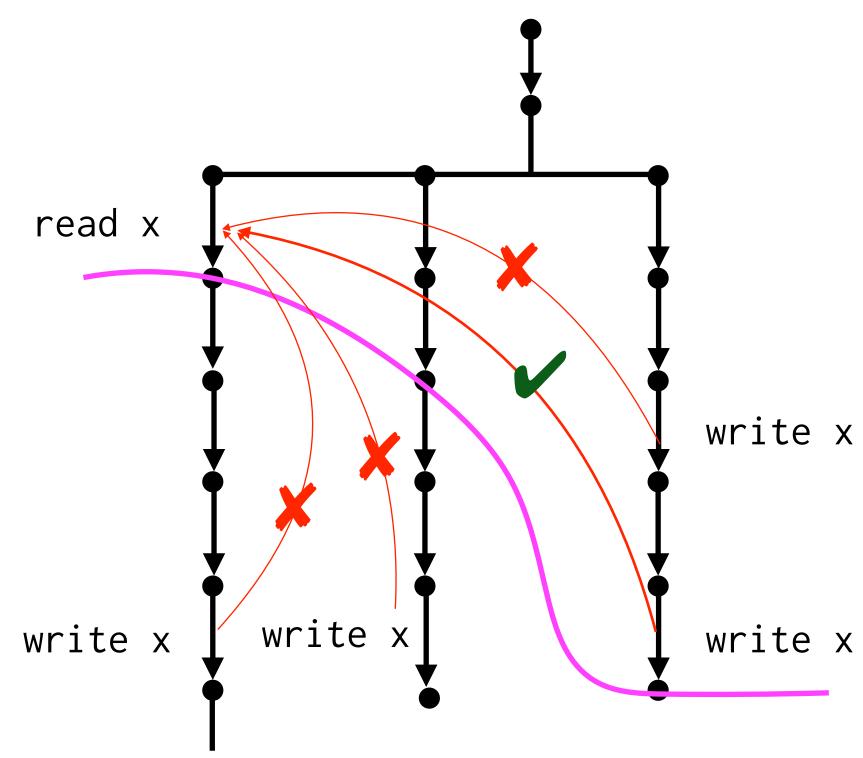


# The weak consistency model

Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017

### Weak consistency models

- Put restrictions on the read-from relation rf
- e.g. sequential consistency: a read at a cut reads from that last write in a process before that cut





Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017

Difficulties

### Naming entities

- Invariants are logical formulæ
- can only describe entities that they name
- in invariants

### • L/O-G use the name of shared variables to designate their current value



## Naming entities

- Invariants are logical formulæ
- can only describe entities that they name
- in invariants

of a shared variable"

• L/O-G use the name of shared variables to designate their current value

### Difficulty

Meaningless with WCMs since there is no notion of ``the current value



### What is known on communications?

- read
- Need to be named  $\rightarrow$  Pythia Variables

• Each process only knows the value of the shared variables from its last

### What we know on communications?

- read
- Need to be named  $\rightarrow$  Pythia Variables

Difficulty

- Its dynamic, not static!
- executed  $\rightarrow$  Stamps (abstraction of local time)

• Each process only knows the value of the shared variables from its last

• A program read action can read from a different write each time it is

# Back to the anarchic semantics

Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017



### • Per process:

- A stamp (local time, no global time)
- A program counter
- The value of the local variables (registers) of the process
- The stamped pythia variables (uniquely identifying <u>all</u> reads along a trace)
- The value of the pythia variables (what was read)
- The read-from relation (rf)

### Example (Peterson)

 $0:\{ w F1 false; w F2 false; w T 0; \}$ P0: 1:w[] F1 true 2:w[] T 2 3:do  $\{i\}$ 7:skip (\* CS1 \*) 8:w[] F1 false Stamps (loop counters)

P1: ||10:w[] F2 true; 11:w[] T 1; 12:do  $\{j\}$ 16:skip (\* CS2 \*) |17:w[] F2 false; Stamps (on loop exit)

### Example (Peterson)

 $0:\{ w F1 false; w F2 false; w T 0; \}$ P0: 1:w[] F1 true 2:w[] T 2 3:do  $\{i\}$ 7:skip (\* CS1 \*) 8:w[] F1 false

### Stamps (loop counters)

P1: 10:w[] F2 true; |11:w[] T 1; 12:do  $\{j\}$ 4: r[] R1 F2 { $\rightarrow$  F2<sup>i</sup><sub>4</sub>} || 13: r[] R3 F1; { $\rightarrow$  F1<sup>j</sup><sub>13</sub>} 5: r[] R2 T { $\rightarrow$  T<sup>i</sup><sub>5</sub>} || 14: r[] R4 T; { $\rightarrow$  T<sup>j</sup><sub>14</sub>} 6:while R1  $\land$  R2  $\neq$  1  $\{i_{end}\}$  | 15:while R3  $\land$  R4  $\neq$  2;  $\{j_{end}\}$ 16:skip (\* CS2 \*) 17:w[] F2 false; Pythia variables Stamps (on loop exit)

Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017

The abstraction

### • For each process

Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017

### • For each process

• For each program point of that process

### • For each process

- For each program point of that process
  - For each execution of the program

- For each process
  - For each program point of that process
    - For each execution of the program
      - For each cut of that execution going through the program point of that process

- For each process
  - For each program point of that process
    - For each execution of the program
      - For each cut of that execution going through the program point of that process

collect:

- For each process
  - For each program point of that process
    - For each execution of the program
      - For each cut of that execution going through the program point of that process
        - collect:
        - The states of all processes, and

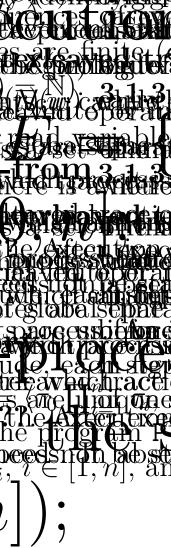
© J. Alglave & P. Cousot

- For each process
  - For each program point of that process
    - For each execution of the program
      - For each cut of that execution going through the program point of that process
        - collect:
        - The states of all processes, and
        - The read-from relation rf

### © J. Alglave & P. Cousot

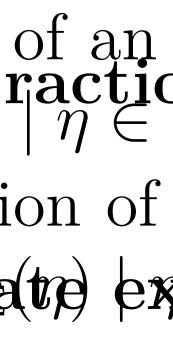
$$\begin{array}{c} \text{ an interfease } \text{ Account of the second of the$$

The candidate & P. Cousot





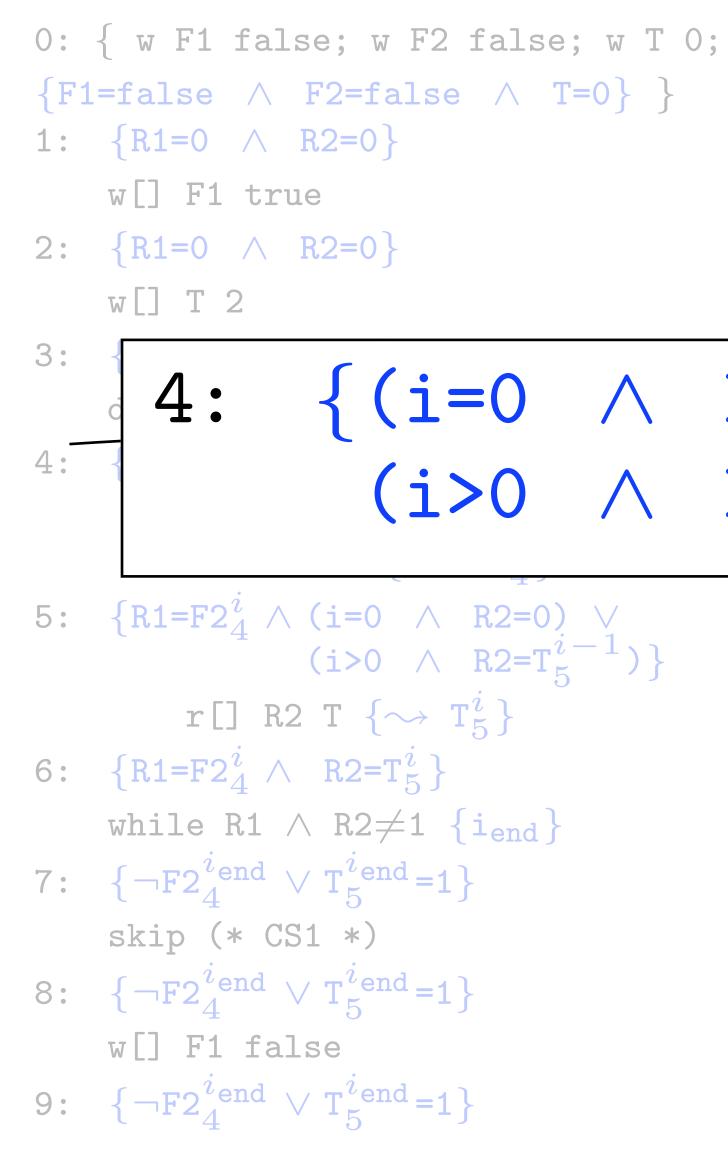






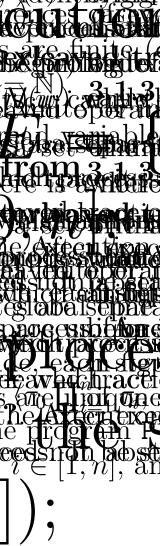


### Example



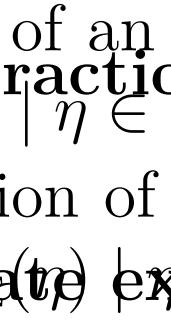
origin of the event Befor3/3threy below the regiment	A stelle realized to proceed to the second to be the process of the proces of the process of the process of the proces of the process of the
<pre>Befor8/alter before / an interleaveer leave of the second of the se</pre>	
; } in figure in fig	ALE BERTER ALE STATES AND A STATES OF A ST
<pre>in figure figure fier extent of one process? In absend of one process? In absend where states are so an interleaved track w[] F2 true; before/after which 11: {R3=0 ^ R4=0} in figure ??. After of one process. In w[] T 1;</pre>	a figure After execution of the prelug
$11: \{R3=0 \land R4=0\}^{in figure ??. After of one process. In figure ??. In figure ??. After of one process. In figure ??.$	f one process. In absence of loss revents the preluce, each stepres the absence of loops events are unconcesses $r_i$ ,
w[] T 1;	$\underline{} i \in [1, n]$
R1=0 $\wedge$ R2=0) $\vee$	i - 1
$R1=F2_{4}^{i-1} \land R2=$	
	<b>1</b> 5 <b>/ /</b>
14: {R3=F1 $_{13}^{j}$ $\land$ (j=0 $\land$ R4=0)	5 $J3.4.0$ • Abstract
$14: \{R3=F1_{13}^{j} \land (j=0 \land R4=0) \land (j>0 \land R4=T_{14}^{j-1}) \land (j=0 \land R4=T_{14}^{j-1}) \land (j>0 $	5 $J3.4.0$ • Abstract
14: {R3=F1 $_{13}^{j}$ $\land$ (j=0 $\land$ R4=0) $\land$ (j>0 $\land$ R4=T $_{14}^{j-1}$	5 $J3.4.0$ • Abstract
$14: \{R3=F1_{13}^{j} \land (j=0 \land R4=0) \land (j>0 \land R4=T_{14}^{j-1}) \\ r[] R4 T; \{ \rightsquigarrow T_{14}^{j} \} \\ 15: \{R3=F1_{13}^{j} \land R4=T_{14}^{j}) \} \\ while R3 \land R4 \neq 2 \{j_{end}\}; \\ 16: \{ \neg F1_{13}^{j_{end}} \lor T_{14}^{j_{end}}=2 \}$	<b>5</b> <b>5</b> <b>5</b> <b>5</b> <b>5</b> <b>5</b> <b>6</b> <b>7</b> <b>7</b> <b>7</b> <b>7</b> <b>7</b> <b>7</b> <b>7</b> <b>7</b>
$14: \{R3=F1_{13}^{j} \land (j=0 \land R4=0) \land (j>0 \land R4=T_{14}^{j}) \\ r[] R4 T; \{ \rightsquigarrow T_{14}^{j} \} \\ 15: \{R3=F1_{13}^{j} \land R4=T_{14}^{j}) \} \\ while R3 \land R4 \neq 2 \{j_{end}\}; \\ 16: \{ \neg F1_{13}^{j_{end}} \lor T_{14}^{j_{end}}=2 \} \\ skip (* CS2 *) \\ 17: \{ \neg F1_{13}^{j_{end}} \lor T_{14}^{j_{end}}=2 \} \\ \end{cases}$	<b>5</b> <b>3.4.0</b> • Abstraction ( The abstraction ( $\alpha_{\Xi}(H) \stackrel{\bullet}{=} \{ \alpha_{\Xi}(\eta) \}$
$\begin{array}{c} 14: \{ \text{R3}=\text{F1}_{13}^{j} \land (j=0 \land \text{R4}=0) \land (j>0 \land \text{R4}=\text{T}_{14}^{j-1} \\ r[] \text{ R4 T}; \{ \rightsquigarrow \text{T}_{14}^{j} \} \\ 15: \{ \text{R3}=\text{F1}_{13}^{j} \land \text{R4}=\text{T}_{14}^{j} \} \\ \text{while R3} \land \text{R4}=\text{T}_{14}^{j} ) \} \\ \text{while R3} \land \text{R4}\neq 2 \{ \text{jend} \} ; \\ 16: \{ \neg \text{F1}_{13}^{j\text{end}} \lor \text{T}_{14}^{j\text{end}}=2 \} \\ \text{skip } (* \text{ CS2 } *) \\ 17: \{ \neg \text{F1}_{13}^{j\text{end}} \lor \text{T}_{14}^{j\text{end}}=2 \} \\ w[] \text{ F2 false}; \end{array}$	<b>5</b> <b>3.4.0</b> • Abstraction of the abstraction of $\alpha_{\Xi}(H) \stackrel{e}{=} \{ \begin{array}{c} \alpha_{\Xi}(\eta) \\ \alpha_{\Xi}(\eta) \end{array}$ The abstraction of the abstr
$14: \{R3=F1_{13}^{j} \land (j=0 \land R4=0) \land (j>0 \land R4=T_{14}^{j}) \\ r[] R4 T; \{ \rightsquigarrow T_{14}^{j} \} \\ 15: \{R3=F1_{13}^{j} \land R4=T_{14}^{j}) \} \\ while R3 \land R4 \neq 2 \{j_{end}\}; \\ 16: \{ \neg F1_{13}^{j_{end}} \lor T_{14}^{j_{end}}=2 \} \\ skip (* CS2 *) \\ 17: \{ \neg F1_{13}^{j_{end}} \lor T_{14}^{j_{end}}=2 \} \\ \end{cases}$	<b>5</b> <b>5</b> <b>5</b> <b>5</b> <b>5</b> <b>5</b> <b>6</b> <b>7</b> <b>7</b> <b>7</b> <b>7</b> <b>7</b> <b>7</b> <b>7</b> <b>7</b>

The candidate & P. Cousot













# The calculational design of the verification conditions by abstract interpretation

Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017

# The induction principle

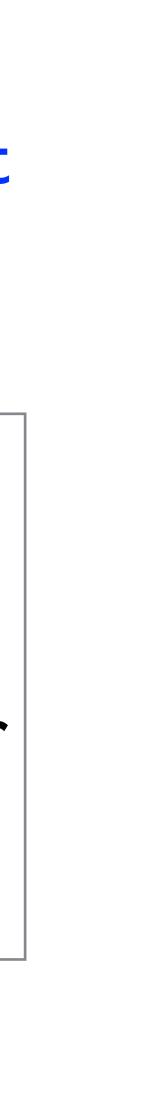
- Given an invariance specification  $S_{inv}$  find a stronger inductive invariant Sind
- Prove that S<sub>ind</sub> satisfy verification conditions
  - Holds after initialization
  - Remains true after a computation step
  - Remains true after a communication
- Assuming S<sub>com</sub> / H<sub>com</sub>

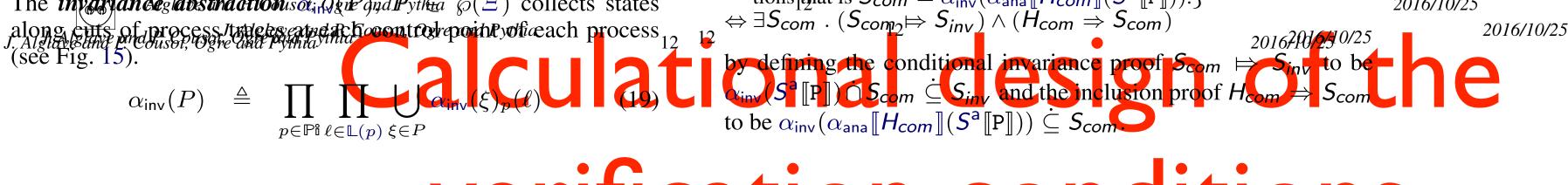
# The induction principle

- Sind
- Prove that S<sub>ind</sub> satisfy verification conditions
  - Holds after initialization
  - Remains true after a computation step
  - Remains true after a communication
- Assuming S<sub>com</sub> / H<sub>com</sub>

• Given an invariance specification  $S_{inv}$  find a stronger inductive invariant

Verification conditions = abstraction of the concrete transformer for one computation step





J. Alglave and P. Cousot, Ogre and Pythia

## verification conditions

- $\alpha_{\mathsf{inv}}(\alpha_{\mathsf{ana}}\llbracket H_{\mathsf{com}} \rrbracket (S^{\mathsf{a}}\llbracket \mathsf{P} \rrbracket)) \subseteq$  $\Leftrightarrow \alpha_{\mathsf{inv}}(\{\xi \in S^{\mathsf{a}}\llbracket \mathsf{P}\rrbracket \mid S\llbracket H_{com}\rrbracket)$  $\Leftrightarrow \alpha_{\mathsf{inv}}(S^{\mathsf{a}}\llbracket \mathbb{P}\rrbracket \cap \{\xi \in S^{\mathsf{a}}\llbracket \mathbb{P}\rrbracket \mid$  $\Leftrightarrow \alpha_{inv}(S^{a}[P]) \cap \alpha_{inv}(\{\xi \in \Xi\})$
- $\Leftrightarrow \alpha_{\rm inv}(S^{\rm a}[\![ \mathbf{P} ]\!]) \cap \alpha_{\rm inv}(\alpha_{\rm ana}[\![ \mathbf{H}_{\alpha}]\!])$  $\Leftrightarrow \exists S_{com} \, . \, \alpha_{inv}(S^{a}[\![\mathbf{P}]\!]) \dot{\cap} S_{com}$  $\mathcal{I}(\Leftarrow)$  For soundness, we have  $\dot{\subseteq} \alpha_{\mathsf{inv}}(S^{\mathsf{a}}[\![\mathsf{P}]\!]) \dot{\cap} S_{\mathit{com}} \dot{\subseteq} S_{\mathit{inv}}$
- tions that is  $S_{com} = \alpha_{inv}(\alpha_{ana} \llbracket H_{com} \rrbracket (S^a \llbracket P \rrbracket))$ .  $\Leftrightarrow \exists S_{com} \ . \ (S_{com} \Rightarrow S_{inv}) \land (H_{com} \Rightarrow S_{com})$

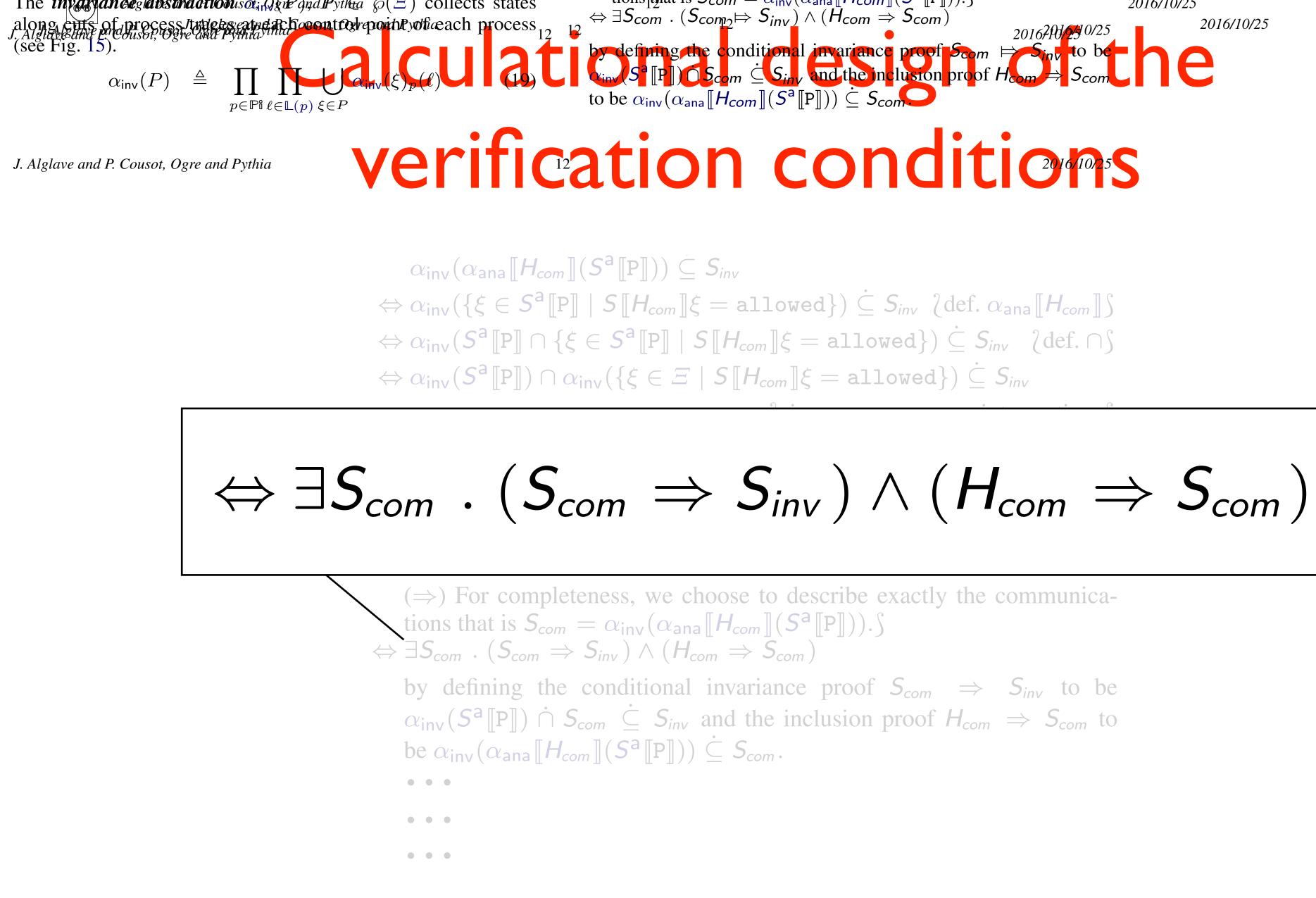
be  $\alpha_{inv}(\alpha_{ana} \llbracket H_{com} \rrbracket (S^a \llbracket P \rrbracket)) \subseteq S_{com}$ .

- • •
- • •
- • •

$$\begin{cases} S_{inv} \\ \xi = \text{allowed} \} ) \subseteq S_{inv} \ (\text{def. } \alpha_{\text{ana}} \llbracket H_{com} \rrbracket ) \\ S \llbracket H_{com} \rrbracket \xi = \text{allowed} \} ) \subseteq S_{inv} \ (\text{def. } \cap ) \\ | S \llbracket H_{com} \rrbracket \xi = \text{allowed} \} ) \subseteq S_{inv} \\ (\text{since } \alpha_{\text{inv}} \text{ preserves intersections}) \\ com \rrbracket (S^{a} \llbracket P \rrbracket) ) \subseteq S_{inv} \ (\text{def. } \alpha_{\text{ana}} \llbracket H_{com} \rrbracket ) \\ \subseteq S_{inv} \land \alpha_{\text{inv}} (\alpha_{\text{ana}} \llbracket H_{com} \rrbracket (S^{a} \llbracket P \rrbracket)) \subseteq S_{com} \\ e \alpha_{\text{inv}} (S^{a} \llbracket P \rrbracket) \cap \alpha_{\text{inv}} (\alpha_{\text{ana}} \llbracket H_{com} \rrbracket (S^{a} \llbracket P \rrbracket)) \\ mv; \end{cases}$$

 $(\Rightarrow)$  For completeness, we choose to describe exactly the communica-

by defining the conditional invariance proof  $S_{com} \Rightarrow S_{inv}$  to be  $\alpha_{inv}(S^{a}[P]) \cap S_{com} \subseteq S_{inv}$  and the inclusion proof  $H_{com} \Rightarrow S_{com}$  to



# Verification conditions

- Sequential proof
- Non-interference proof (like L/O-G but for different kind of invariants)
- Communication proof
  - a read event reading from a write event must be in rf
  - the value read for a variable is the one written
  - reading is fair in rf (cannot be delayed indefinitely)

## (useless in L/O-G since rf is fixed)



# The program consistency hypothesis S<sub>com</sub>

# Communication hypothesis S<sub>com</sub>

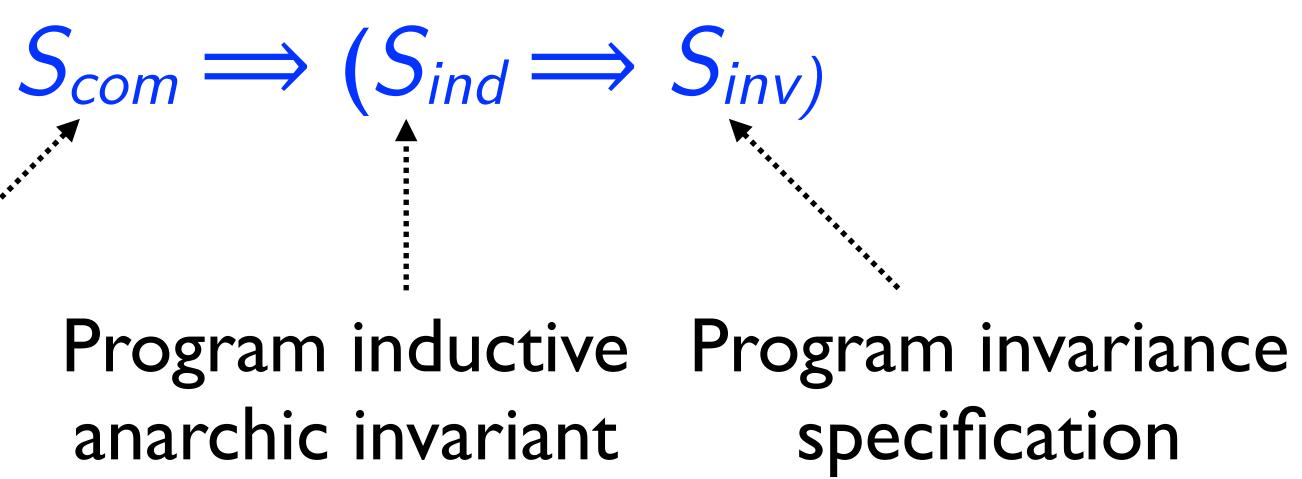
# calculational design:

### Communication hypothesis

## • i.e. (Sind $\land \neg$ Sinv) $\Rightarrow \neg$ Scom • Necessary: by counter examples

Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017

• A sufficient communication hypothesis can be discovered by



# Proving Consistency $H_{com} \implies S_{com}$ $\downarrow$ cat invariant

## Proof method

#### • Obtained by calculational design:

$$\begin{split} &\alpha_{\mathrm{inv}}(\alpha_{\mathrm{ana}}\llbracket H_{\mathrm{com}}\rrbracket(S^{\mathrm{a}}\llbracket \mathbb{P})) \subseteq S_{\mathrm{com}} \\ &\Leftrightarrow \alpha_{\mathrm{inv}}(S^{\mathrm{ana}}\llbracket H_{\mathrm{com}}\rrbracket \mathbb{P}) \subseteq S_{\mathrm{com}} \qquad (\det S^{\mathrm{ana}}\llbracket H_{\mathrm{com}}\rrbracket \mathbb{P}) \\ &\Leftrightarrow \forall \xi \in S^{\mathrm{ana}}\llbracket H_{\mathrm{com}}\rrbracket \mathbb{P} \cdot \alpha_{\mathrm{inv}}(\{\xi\}) \subseteq S_{\mathrm{com}} \qquad (\alpha_{\mathrm{inv}} \operatorname{preserves} \cup) \\ &\Leftrightarrow \forall \xi \in S^{\mathrm{ana}}\llbracket H_{\mathrm{com}}\rrbracket \mathbb{P} \cdot \bigcup_{p=1}^{n} \bigcup_{\substack{i \in \mathbb{P}_p}} \{\alpha_{\mathrm{inv}}(\xi')_p(\mathbf{L}) \mid \xi' \in \{\xi\}\} \subseteq S_{\mathrm{com}} \\ &\qquad (\det (19) \text{ of } \alpha_{\mathrm{inv}}) \\ &\Leftrightarrow \forall (\tau_{\mathrm{start}} \times \prod_{p=0}^{n-1} \tau_p \times \pi \times \mathrm{rf}) \in S^{\mathrm{ana}}\llbracket H_{\mathrm{com}}\rrbracket \mathbb{P} \cdot \forall p \in [1, n] \cdot \forall \mathbf{L} \in \mathbb{P}_p \\ &\qquad \alpha_{\mathrm{inv}}(\tau_{\mathrm{start}} \times \prod_{p=0}^{n-1} \tau_p \times \pi \times \mathrm{rf})_p(\mathbf{L}) \subseteq S_{\mathrm{com}_p}(\mathbf{L}) \\ &\qquad (\det , \in, \bigcup, \subseteq, \text{ and } S^{\mathrm{ana}}\llbracket H_{\mathrm{com}}\rrbracket \mathbb{P} \text{ so that } \xi \text{ has the form } \xi = \\ &\qquad \tau_{\mathrm{start}} \times \prod_{p=0}^{n-1} \tau_p \times \pi \times \mathrm{rf} \\ &\qquad p \in \mathfrak{f} \\ &\Leftrightarrow \forall (\tau_{\mathrm{start}} \times \prod_{p=0}^{n-1} \tau_p \times \pi \times \mathrm{rf}) \in S^{\mathrm{ana}}\llbracket H_{\mathrm{com}}\rrbracket \mathbb{P} \cdot \forall i \in (20) \\ &\qquad (1, n] \cdot \forall \mathbf{L} \in \mathbb{P}_p \cdot \forall q \in [0, n[ \cdot \forall k_q < |\tau_q| \cdot (\frac{\tau_q}{k_q} = s\langle \kappa_q, k_q, \theta_q, k_q, \rho_q, k_q, \nu_q, k_q \rangle \wedge \kappa_p, k_p = \mathbf{L}) \Rightarrow \\ &\qquad \langle \kappa_{0, k_0}, \theta_{0, k_0}, \rho_{0, k_0}, \dots, \nu_{p-1, k_{p-1}}, \theta_{p, k_p}, \rho_{p, k_p}, \nu_{p, k_p}, \\ &\qquad \kappa_{p+1, k_{p+1}, \dots, \kappa_{n-1, k_{n-1}}, \theta_{n-1, k_{n-1}}, \rho_{n-1, k_{n-1}}, \nu_{n-1, k_{n-1}}, \mathrm{rf} \\ &\in S_{\mathrm{com}_i}(\mathbf{L}) \\ \end{aligned}$$

## Proof method

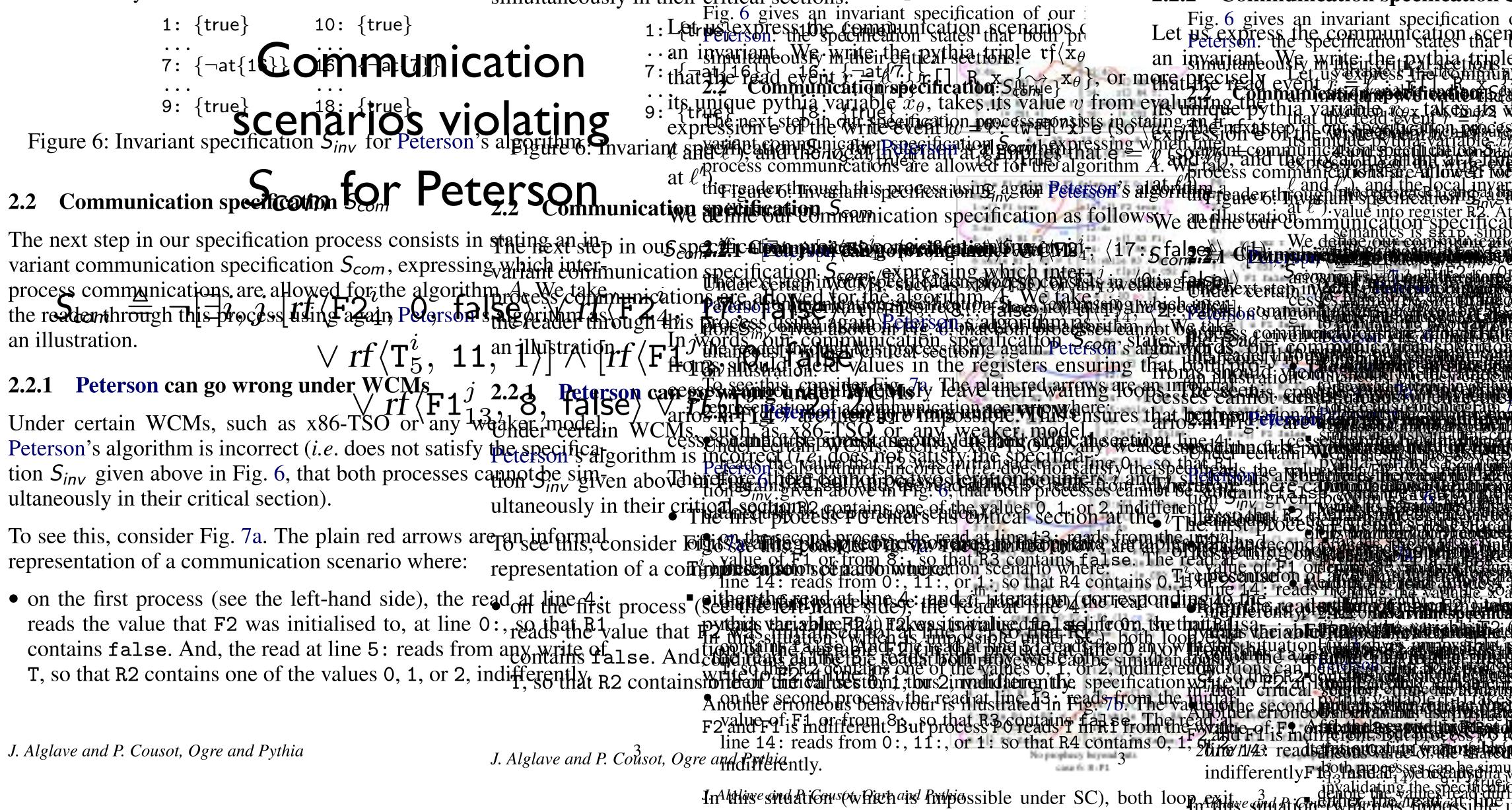
- The anarchic invariants can be used to calculate all communication scenarios violating  $S_{com}$
- These scenarios must be forbidden by the cat specification H<sub>com</sub>

(no need to reason at the level of traces of the anarchic semantics)

#### **2.1.2** Invariant specification $S_{inv}$

#### **2.1.2** Invariant specification S<sub>inv</sub>

Fig. 6 gives an invariant specification of our implementation of an invariant specification of our implementation of Peterson: the specification states that both processes cannot be simultaneously in their critical sections. Simultaneously in their critical sections. Simultaneously in their critical sections.



Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017

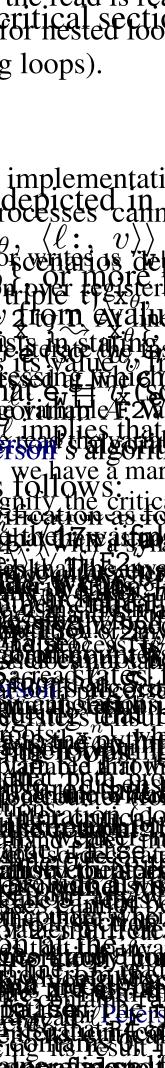
 $\cdot \cdot$  an invariant of  $x_{\theta}$  write in the text beautriple  $rf(x_{\theta})$ 

both processes can be simultaneously in their critical section invalidating the space of eating soft all surrounding loops).

#### 2.2.2.1.2 Invariant specification $S_{invitron}$

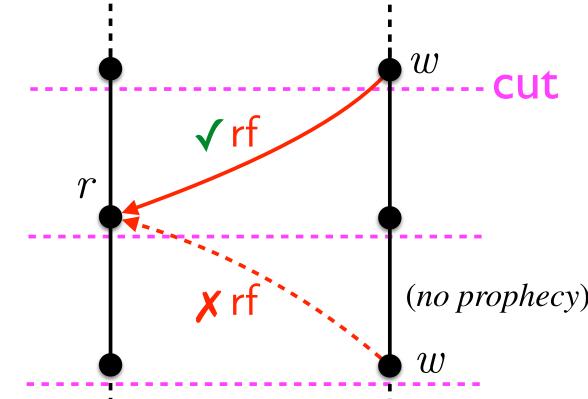
Fig. 6 gives an invariant specification of our 1: Letrus expression communication scenarios ( Fig. 6 gives an invariant specification of our implementation Let us express the communication scenarios depicted in peterson. the specification states that both processes can an invariant  $We write the pythia triple <math>rf(\bar{x}_{\theta}, \langle \ell:, v \rangle)$ 7: that the fead event  $\{-at_{\ell}/2, \{-n\}, n\}$ , or make the isotropy of the communication specification:  $\{x_{\theta}\}$ , or make the isotropy of the event  $\{x_{\theta}\}$  its unique pythia variable  $x_{\theta}$ , takes its value v from evaluating the type of the type of the pythia variable  $x_{\theta}$ , takes its value v from evaluating the pythia variable  $x_{\theta}$ , takes its value v from evaluating the pythia variable  $x_{\theta}$ , takes its value v from evaluating the pythia variable  $x_{\theta}$  takes its value v from evaluating the pythia variable  $x_{\theta}$ , takes its value v from evaluating the pythia variable  $x_{\theta}$ , takes its value v from evaluating the pythia variable  $x_{\theta}$  takes its value v from evaluating the pythia variable  $x_{\theta}$  takes its value v from evaluating the pythia variable  $x_{\theta}$  takes its value v from evaluating the pythia variable  $x_{\theta}$  takes its value v from evaluating the pythia variable  $x_{\theta}$  takes its value v from evaluating the pythia variable  $x_{\theta}$  takes its value v from evaluating the pythia variable  $x_{\theta}$  takes its value v from evaluating the pythia variable  $x_{\theta}$  takes its value v from evaluating the pythia variable  $x_{\theta}$  takes its value v from evaluating the pythia variable  $x_{\theta}$  takes its value v from evalue v from eval 9: {true} 9: {true} Scenarios violating Figure 6: Invariant specification S<sub>inv</sub> for Peterson's algorithms. Invariant appendix and the process communications are allowed for the algorithm A. Werders communication between the the difference of the algorithm A. Werders communications are allowed for the algorithm A. the reader the local invariant at limplies that the reader the local invariant at limplies that and li The next step in our specification process consists in stating an inp in our specification specifica variant communication specification  $S_{com}$ , expressing which inter-index established for the algorithm A. We take the second single specification in the second sec process communications are allowed for the algorithm  $A_{c}$  we take in the reader through this process communication are allowed for the algorithm  $A_{c}$  we take in the reader through this process lising again Poerson showing the process lising again process lising ag 2.2.1 Peterson can go wrong under WCMs j 2.2.8 Peterson cacego wr ultaneously in their critical section). To see this, consider Fig. 7a. The plain red arrows are an informal consider Fig. 2 information of a communication scenario where: • on the first process (see the left-hand side), the read of line first process (see the left-hand side) is the line first process (see the left-hand side) is the line first process (see the left-hand side) is the line first process (see the left-hand side) is the line first p reads the value that F2 was initialised to, at line 0:, so that B1 value that F2 was site to be both loop years the both loop Another encoded storn stors international stores in the second of the se ZUTTER 11/43. reads and the second by the second se indifferentlyF189tharears sector absolution warranges with the give u

In Addise site attion with and Propossible under SC), both loop exit 3 in the site attion of the specification Scomputation. (Which and Propossible under SC), both loop exit 3 in the site attion of the specification of



## Incompleteness

- Sinv, Sind, and Scom remain valid)
- $S_{com}$  can refer to communicated values not  $H_{com}$  in cat know about communicated values)
- cat may not be expressive enough:

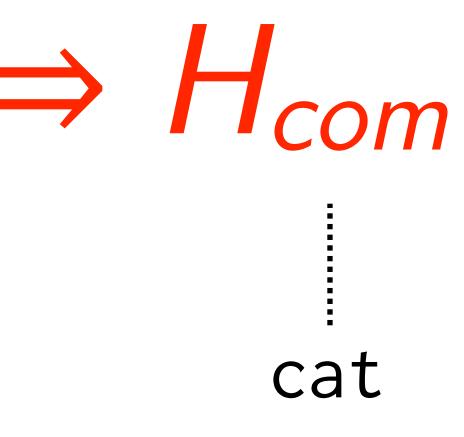


### • In general you have to add fences for $H_{com}$ (do not change the invariants,

# (redesign your algorithm without assuming that the hardware does

### No read beyond cut

# Proving Architectural Consistency $M \Longrightarrow H_{com}$ cat cat



# $M \Longrightarrow H_{com}$ in cat

#### sound and complete proof method

### unpublished paper of JA and PC with Luc Maranget

# Beyond L/O-G: non-starvation

# Reasoning on one execution only

- relation rf
- Not directly possible with L/O-G
- Can be used to prove non-starvation

• A particular execution can be uniquely characterized by its read-from

## • We can reason on one execution only $(S_{com}$ for this execution + $S_{ind})$

# Non-starvation (e.g. PostgrSQL)

- Consider all traces that may star trace)
- Prove each of them to be infeasible:
  - the inductive invariant  $S_{ind}$  under the program communication hypothesis  $S_{com}$  is unsatisfied
  - or, by strengthening the program communications  $S_{com}$  (maybe implemented by adding fences in  $H_{com}$ )

• or, by a fairness hypothesis.

#### • Consider all traces that may starve (for an appropriate S'<sub>com</sub> for each

# Communication fairness hypothesis®

- All writes eventually hit the memory:
  - If, at a cut of the execution, all the processes infinitely often write the same value  $\upsilon$  to a shared variable x and only that value  $\upsilon$
  - and from a later cut point of that execution, a process infinitely often repeats reads to that variable x
  - then the reads will end up reading that value  $\upsilon$

(\*) The SPARC Architecture Manual, Version 8, Section K2, p. 283: ``if one processor does an S, and another processor repeatedly does L 's to the same location, then there is an L that will be after the S".



Ogre and Pythia: an Invariance Proof Method for Weak Consistency Models, POPL 2017, 18-20 January 2017

Conclusion

# Conclusion

- To design a correct parallel algorithm, specify:
  - the algorithm
  - the invariance specification  $S_{inv}$
  - the program-specific consistency model  $S_{com}$
- Find an anarchic inductive invariant  $S_{ind}$  satisfying the verification conditions such that  $(S_{com} \land S_{ind}) \Longrightarrow S_{inv}$

# Conclusion

- To implement a parallel algorithm correctly:
  - Implement the program consistency model on an architecture consistency model M (possibly adding fences)
  - Prove  $M \Longrightarrow S_{com}$
- Or better
  - Find a minimal/weakest  $H_{com}$  such that  $H_{com} \Longrightarrow S_{com}$
  - $M \Longrightarrow H_{com}$



- Specification of parallel/distributed program consistency models (more refined than <u>architecture</u> consistency models, e.g. cuts needed)
- Liveness (beyond non-starvation)
- Collection of certified algorithms for WCM (e.g. transactional memory, databases, etc)
- Static analysis (by abstract interpretation of the analytic semantics parameterized by a WCM)

## More work needed



# The End, Thank You