A generic hardware in-order pipeline architecture model to capture key temporal properties

J.-P. Bodeveix, A. Bonenfant, T. Carle,
M. Filali, C. Rochange, L. Sylvestre
IRIT CNRS-Université de Toulouse FRANCE

TLA+ Community Meeting April 12, 2026
Co-located with ETAPS 2026
in Torino - ITALY

Table of Contents

Plan

Real time systems

- design, validation, certification.
- accurate and *safe* Worst-Case Execution Time (WCET) bounds.
- hardware aspects
  - architectures to make tractable analyses
  - validation of architectural properties

Timing anomalies

- A timing anomaly occurs when the usual assumptions to make analyses tractable are broken.
- examples of usual assumptions:
  - a cache hit leads to a globally faster execution.
  - a cache miss leads to a globally slower execution.
- examples of anomalies:
  - a cache hit leads to a a globally *slower* execution.
  - a cache miss leads to a globally *faster* execution.

  Basic anomalies:
  - counter-intuitive anomaly.
  - amplification anomaly.

Plan

Towards an abstract pipeline model

- System view
- Untimed non-deterministic pipeline model
- Timed deterministic pipeline model
- Basic properties

System view

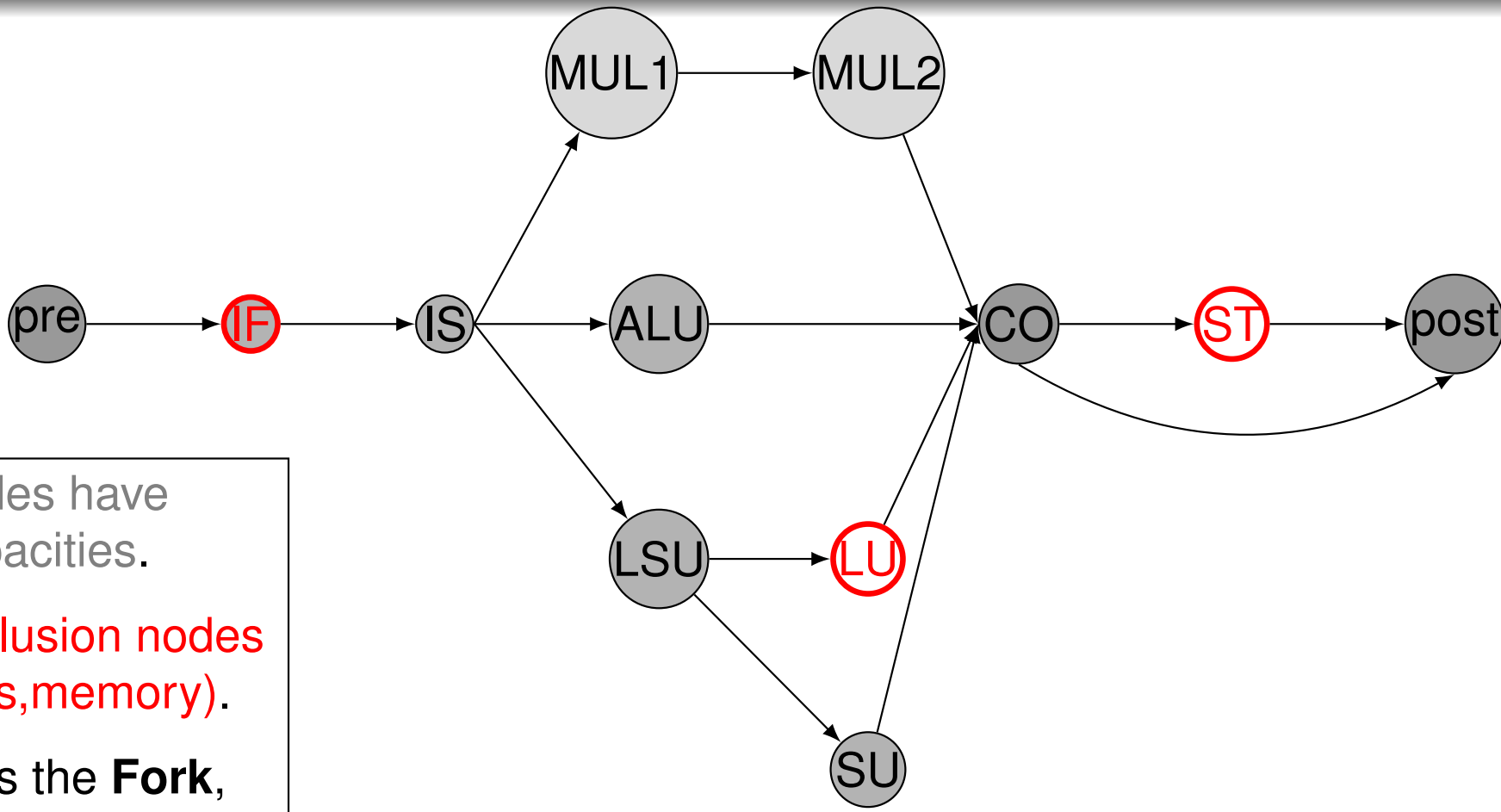
Static architecture features

- The pipeline is an acyclic graph of stages
- Each stage has a capacity and a latency
- Some stages are connected to the memory
- Memory access is exclusive.

Dynamic architecture features

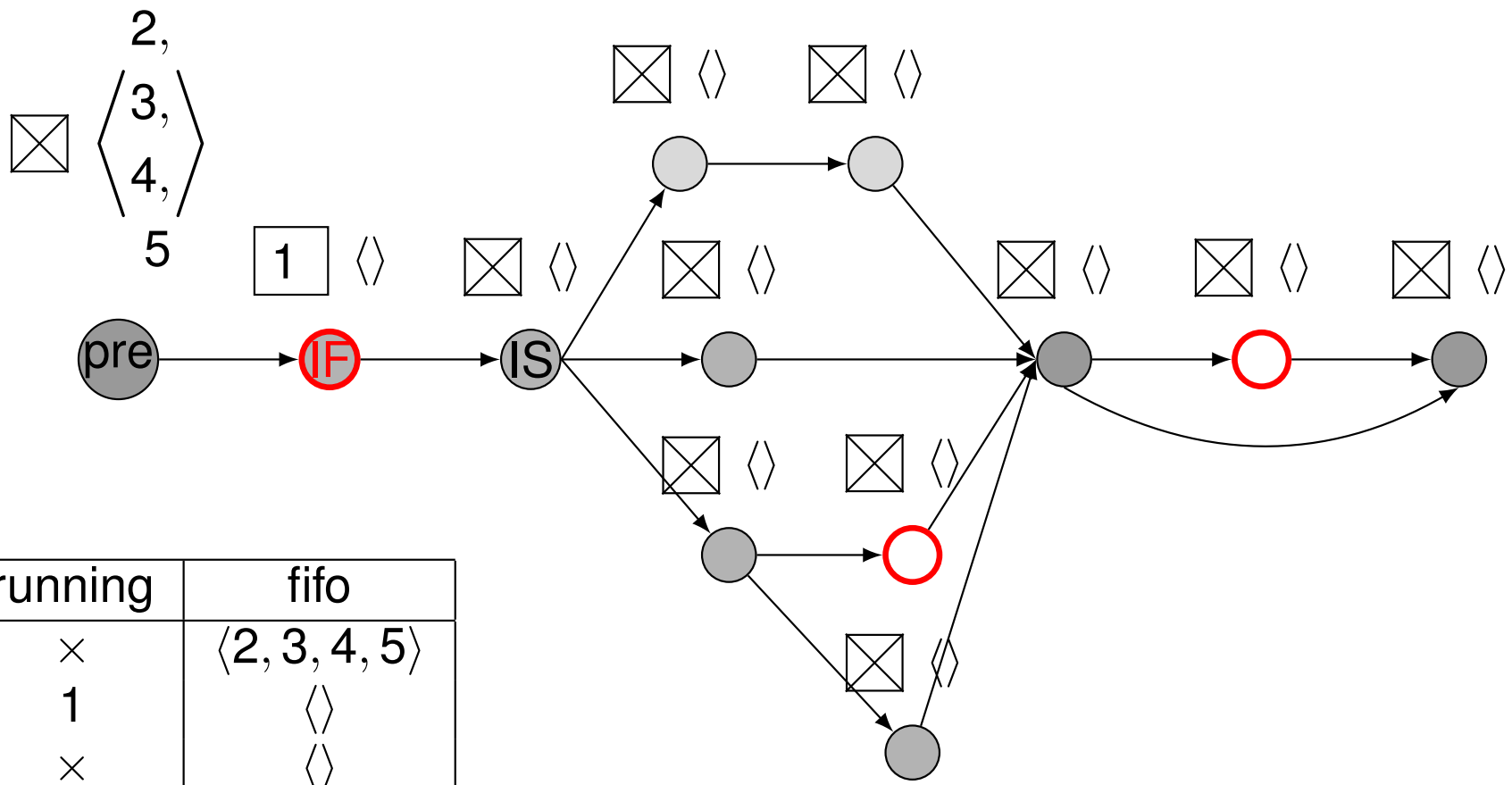
- A program is a sequence of instructions.
- Each instruction follows a path over the pipeline stages.
- Each instruction has a latency on each stage.

Static Architecture features



- nodes have capacities.
- exclusion nodes (bus,memory).
- IS is the **Fork**, CO is the **Join**.

Dynamic architecture features (1)



stage	running	fifo
pre	×	⟨2, 3, 4, 5⟩
IF	1	⟨⟩
IS	×	⟨⟩
...	×	⟨⟩

Dynamic architecture features (2)

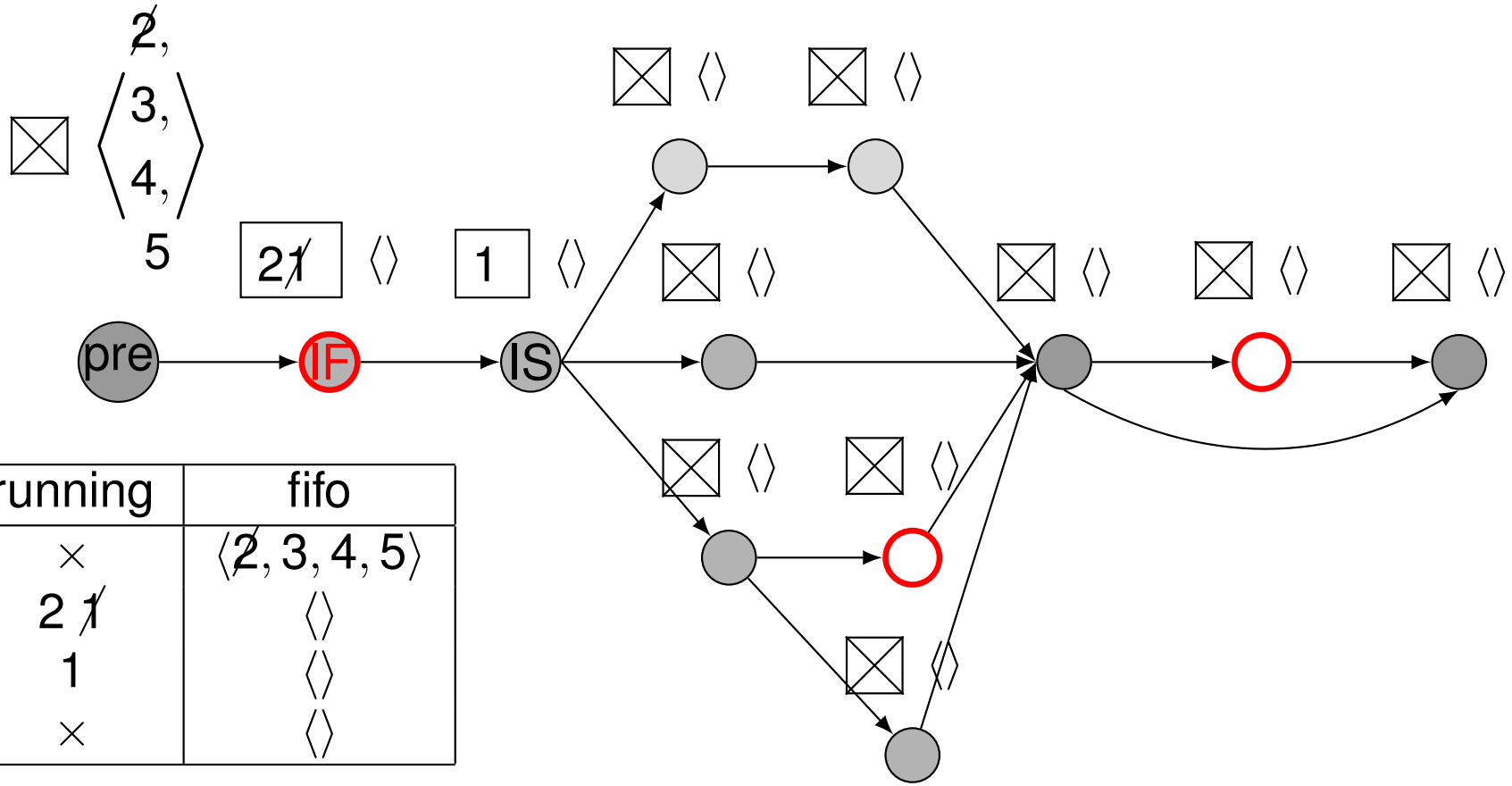
A(?) subset of runnings is selected and free their respective stage either by leaving or by staying. Then, the move has to:

- respect the capacity of stages.
- be in order.
- respect stage exclusion.
- not interrupt the remaining runnings (no preemption).
- be maximal.
- and *deterministic*.

Dynamic architecture features (3)

$\{1\} \subseteq \{1\}$ is selected.

1 leaves IF and becomes running on IS.
2 leaves pre and becomes running on IF.

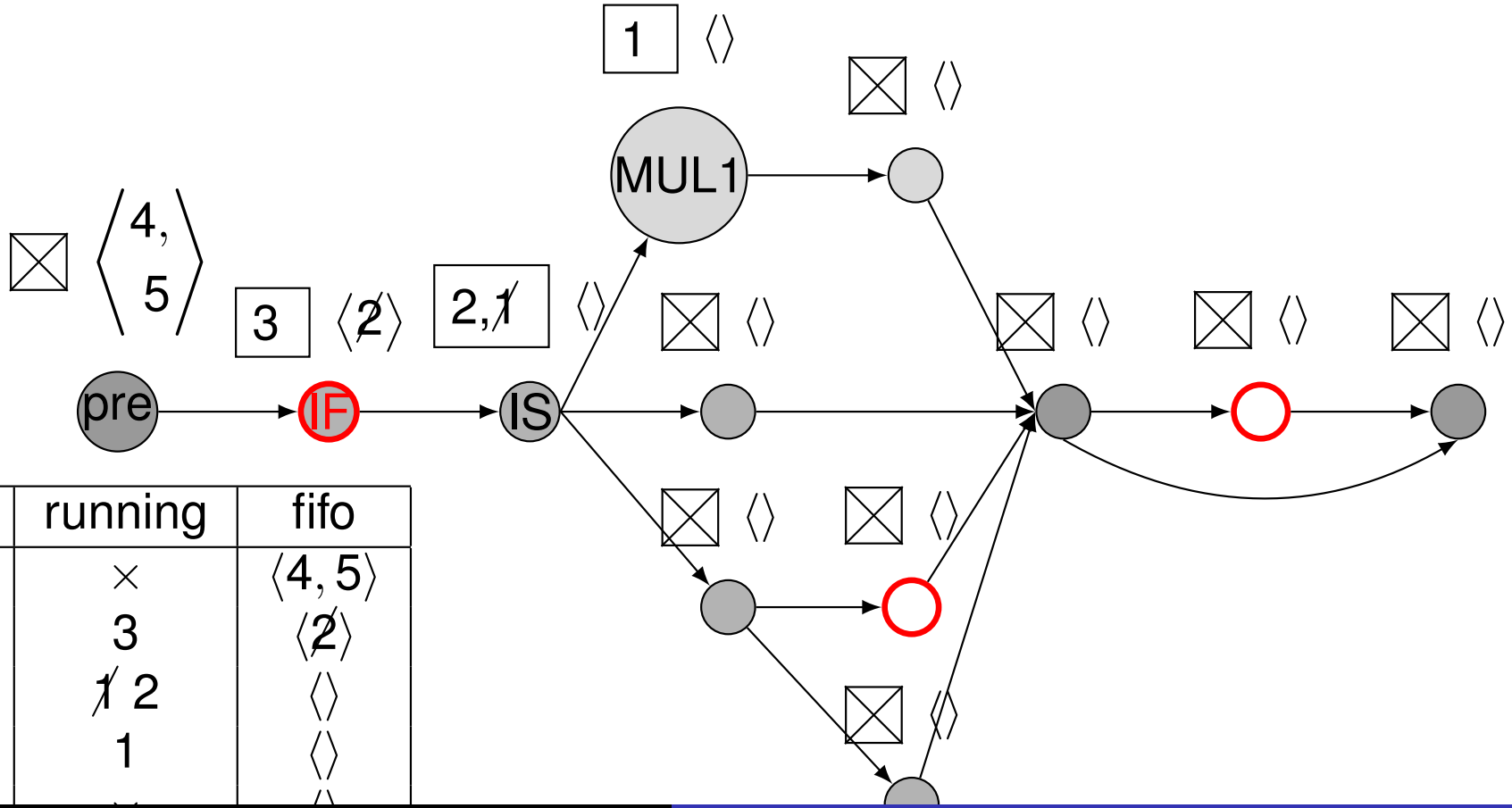


stage	running	fifo
pre	×	$\langle 2, 3, 4, 5 \rangle$
IF	2 1	$\langle\rangle$
IS	1	$\langle\rangle$
...	×	$\langle\rangle$

Dynamic architecture features (4)

$\{1\} \subseteq \{1,3\}$ is selected.

1 leaves IS and becomes running on MUL1.
2 leaves IF and becomes running on IS.



stage	running	fifo
pre	×	⟨4, 5⟩
IF	3	⟨2⟩
IS	1 2	⟨⟩
MUL1	1	⟨⟩

A timed deterministic pipeline model

- A *global clock* counts the processors cycles. Each instruction has a *count*.
- When elected a running has its *count* initialized to its given latency on its current stage. It will be decremented at each tick of the clock.
- **The subset** of selected runnings are exactly those with their *count* equal to 0.

A timed deterministic pipeline model

- The choice is deterministic.
- The move is deterministic.
- $\rightsquigarrow$ The pipeline model is deterministic.

Plan

Pipeline graph normal form

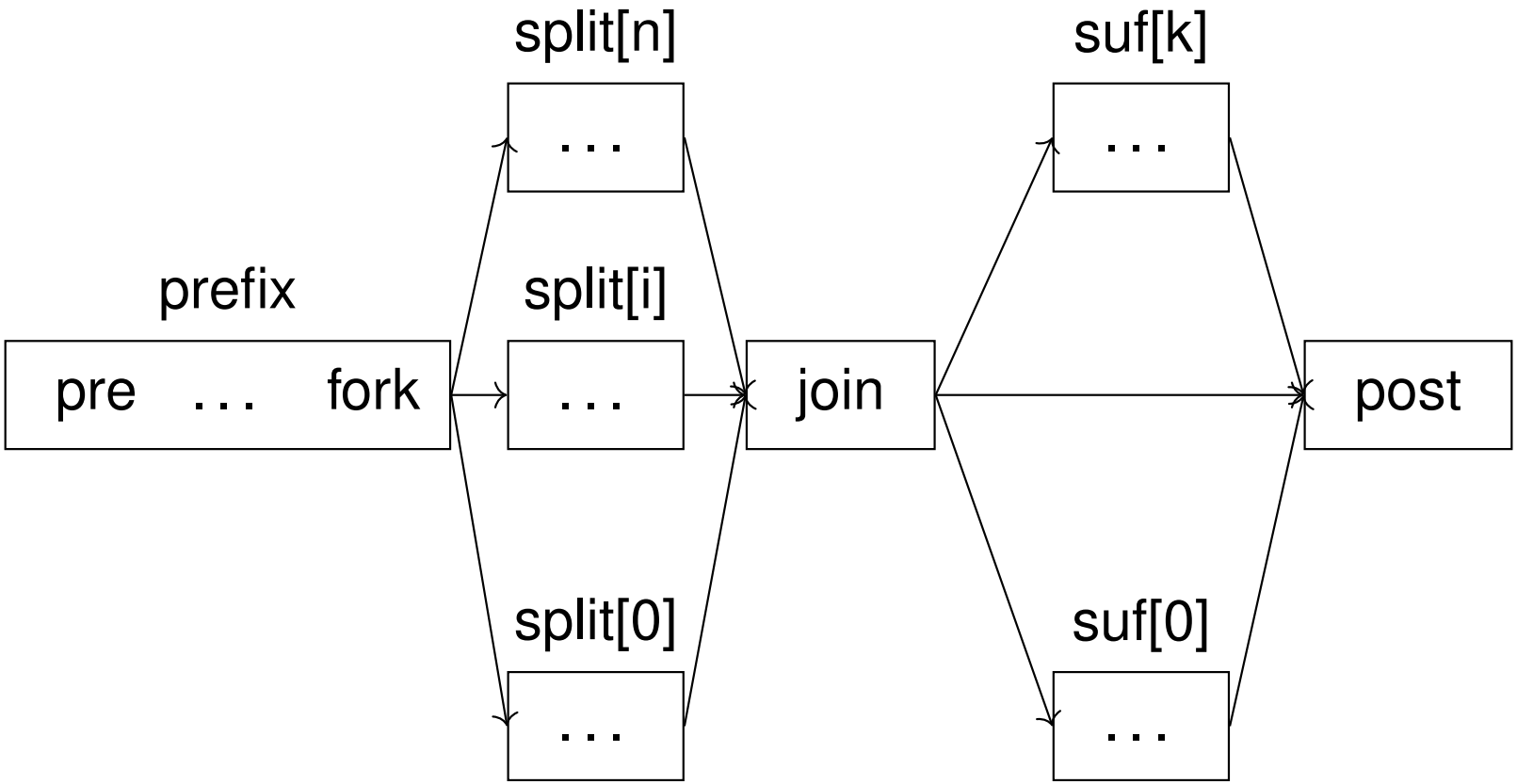


Figure: Pipeline graph normal form.

Path decomposition

$$\text{ASSUME } \mathit{Stages\_partition} \triangleq$$
$$\mathit{Stages} = (\{\mathit{pre}\} \cup \mathit{Range}(\mathit{prefix}) \cup \{\mathit{fork}\}$$
$$\cup (\text{UNION } \{\mathit{Range}(\mathit{spl}[b]) : b \in \text{DOMAIN } \mathit{spl}\})$$
$$\cup \{\mathit{join}\}$$
$$\cup (\text{UNION } \{\mathit{Range}(\mathit{suf}[s]) : s \in \text{DOMAIN } \mathit{suf}\})$$
$$\cup \{\mathit{post}\})$$

$$\mathit{route} \triangleq [\mathit{at\_fork} : \text{DOMAIN } \mathit{spl},\ \mathit{at\_join} : \text{DOMAIN } \mathit{suf},\ \ldots$$

$$\mathit{route2path}(r) \triangleq$$
$$\langle \mathit{pre} \rangle \circ \mathit{prefix} \circ \langle \mathit{fork} \rangle$$
$$\circ\ \ \mathit{spl}[r.\mathit{at\_fork}] \circ \langle \mathit{join} \rangle$$
$$\circ\ \mathit{suf}[r.\mathit{at\_join}] \circ \langle \mathit{post} \rangle$$

Basic traversals

- At each cycle, a move along the pipeline is a **global** operation over stages:
  - stage allocation (willbefree [RH20,GCRCP22])
  - join advance [GCRCP22]
- How to unwind this basic recursive procedure call ?
  $\rightsquigarrow$ study of a global *right to left traversal: from post to pre*

Scoreboard abstraction

- From right to left the traversal is generally deterministic: for each stage, *but the join*, the predecessor is unique.
- The (abstract) role of the scoreboard is to determine which is the *current* predecessor.
  - Each instruction carries a bounded reorder counter ($\neq$ unbounded instruction number).
  - Instructions are tagged at the fork stage and controlled at the join stage.

$$SCB \triangleq [i\_roc : Nat \text{ input reorder counter}, o\_roc : Nat \text{ output reorder counter}]$$

Plan

Pipeline basic properties

$$
\begin{aligned}
&\mathit{Insts\_partition} \triangleq \\
&\quad \land (\text{UNION } \{\mathit{Insts\_on}(\mathit{St}, \mathit{st}) : \mathit{st} \in \mathit{Stages}\}) = \mathit{Insts} \\
&\quad \land \forall\, \mathit{st}1, \mathit{st}2 \in \mathit{Stages} : \\
&\qquad \mathit{Insts\_on}(\mathit{St}, \mathit{st}1) \cap \mathit{Insts\_on}(\mathit{St}, \mathit{st}2) \neq \{\} \implies \mathit{st}1 = \mathit{st}2
\end{aligned}
$$

$$
\mathit{Cap} \triangleq \forall\, \mathit{st} \in \mathit{Stages} : \mathit{Len}(\mathit{St.on}[\mathit{st}].\mathit{fifo}) \leq \mathit{cap}[\mathit{st}]
$$

$$
\begin{aligned}
&\mathit{mem\_access}(\mathit{st}) \triangleq \\
&\quad \land \mathit{st} \in \mathit{StMem} \\
&\quad \land \mathit{St.on}[\mathit{st}].\mathit{running} \in \mathit{Insts} \\
&\quad \land \mathit{st} \notin \mathit{hit}[\mathit{St.on}[\mathit{st}].\mathit{running}]
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{Mutex\_mem} \triangleq \forall\, \mathit{st\_i}, \mathit{st\_j} \in \mathit{Stages} : \\
&\quad \mathit{mem\_access}(\mathit{st\_i}) \land \mathit{mem\_access}(\mathit{st\_j}) \implies \mathit{st\_i} = \mathit{st\_j}
\end{aligned}
$$

Basic safety property (1)

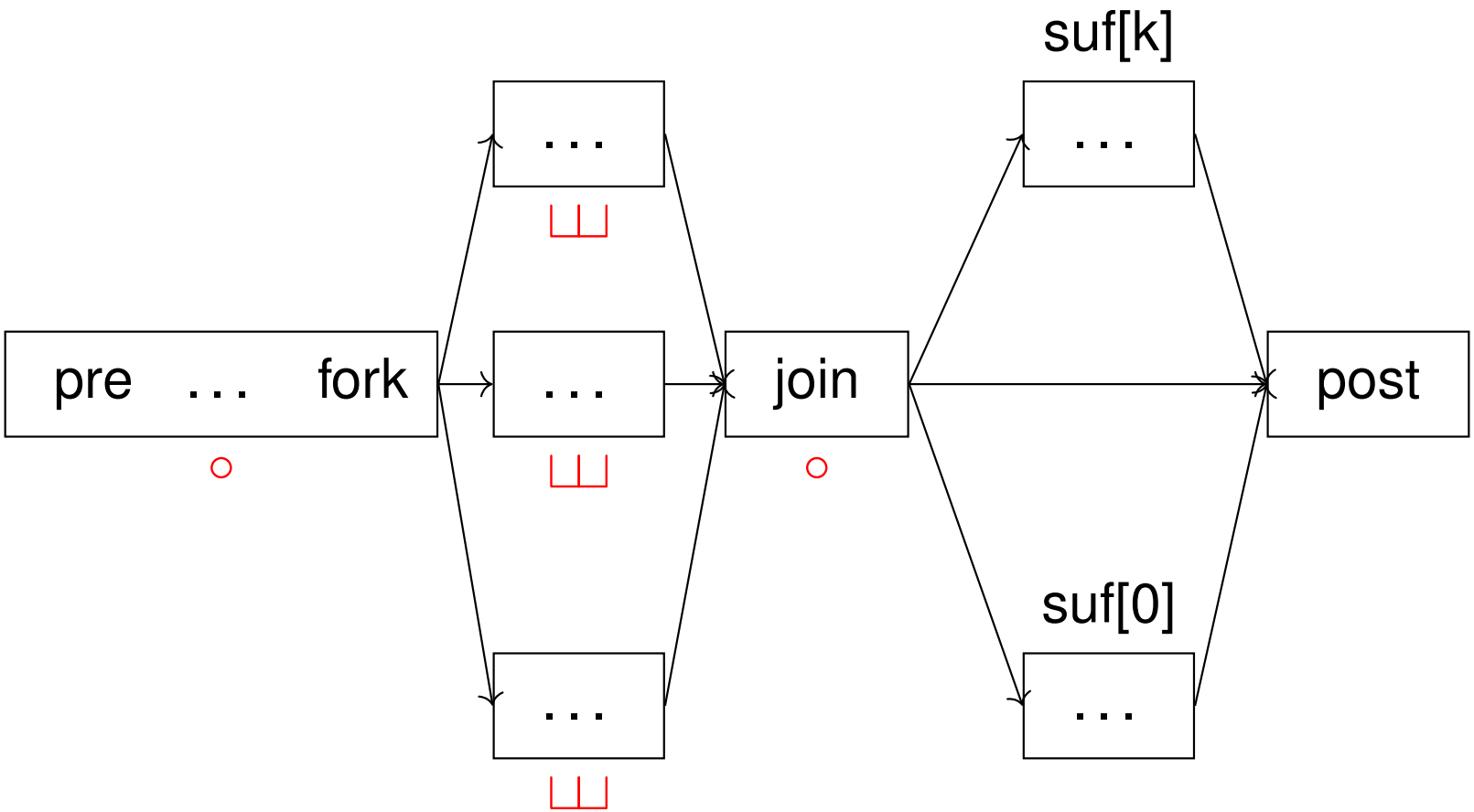


Figure: Auxiliary ordered reconstruction.

Basic safety property (2)

$ordered\_rec \triangleq$ auxiliary ordered reconstruction

$norm(St.on[join].fifo)$ implicit order of the join fifo

$\circ\ spl2fifo(St)$ fifo reconstruction over Split stages thanks to reorder counters provided by the scoreboard abstraction

$\circ\ fifos\_pre\_prefix\_fork(St))$ implicit order over the linear fifos

Basic safety property (3)

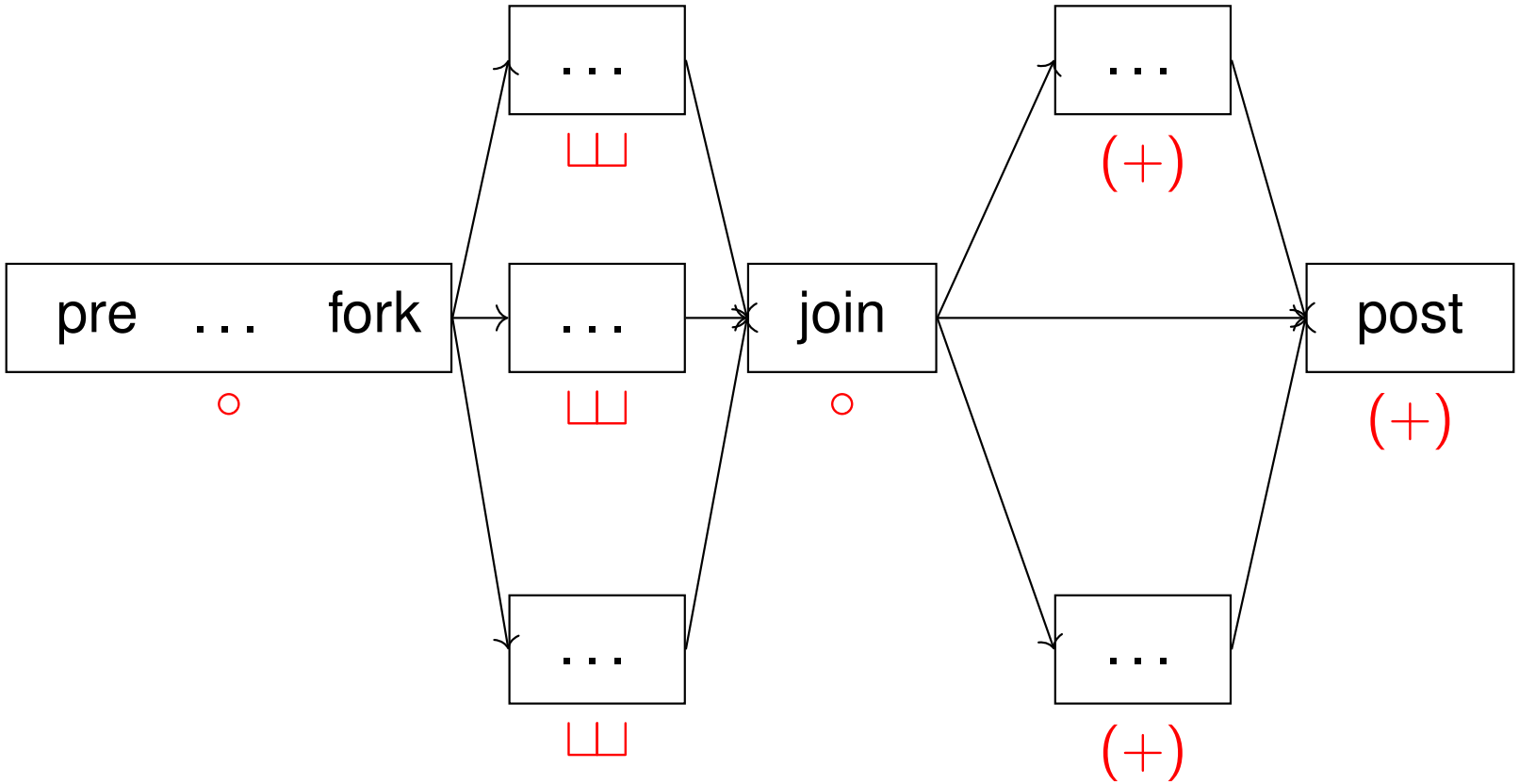


Figure: Full Reconstruction.

Basic safety property (4)

$$
\begin{array}{l}
\mathit{No\_loss} \triangleq \\
\text{LET } \mathit{suf\_stages} \triangleq \text{ UNION } (\{\mathit{Range}(\mathit{suf}[s]) : s \in \text{DOMAIN } \mathit{suf}\})\text{IN} \\
\quad \wedge \mathit{ordered\_rec} = \\
\qquad \mathit{SubSeq}(\mathit{fifo\_Init}, \mathit{Len}(\mathit{fifo\_Init}) - \mathit{Len}(\mathit{ordered\_rec}) + 1, \mathit{Len}(\mathit{fi} \\
\quad \wedge (\mathit{BagUnion}(\{\mathit{ToBag}(\mathit{norm}(\mathit{St.on}[s].\mathit{fifo})) : s \in \mathit{suf\_stages}\}) \\
\qquad \oplus \\
\qquad \mathit{ToBag}(\mathit{norm}(\mathit{St.on}[\mathit{post}].\mathit{fifo})) \\
\qquad = \mathit{ToBag}(\mathit{SubSeq}(\mathit{fifo\_Init}, 1, \mathit{Len}(\mathit{fifo\_Init}) - \mathit{Len}(\mathit{ordered\_rec}))
\end{array}
$$

Basic liveness property

Liveness property: Termination is eventually reached and stable.

$$
\begin{array}{l}
\mathit{pipe\_Termination} \triangleq \\
\quad \Diamond\Box(\land \forall\, \mathit{st} \in \mathit{Stages} : \mathit{st} \neq \mathit{post} \implies \mathit{St.on}[\mathit{st}].\mathit{fifo} = \langle\rangle \quad \text{no garbage} \\
\qquad \land \forall\, \mathit{st} \in \mathit{Stages} : \mathit{St.on}[\mathit{st}].\mathit{cnt} = 0 \quad \text{no interrupt} \\
\qquad \land \mathit{ToBag}(\mathit{St.on}[\mathit{post}].\mathit{fifo}) = \mathit{ToBag}(\mathit{fifo\_Init}) \quad \text{no loss} \\
\quad)
\end{array}
$$

Plan

Conclusion and perspectives

- Development of a TLA$^+$ pipeline model:
  - Fruitful (animated) interaction with the hardware team
    $\leadsto$ better global view of the protocol.
  - TLA$^+$ expressive power
  - Proof engineering in progress
- Future work:
  - More detailed mechanism modeling (e.g., scoreboard)
  - Refinement toward implementation-level model
  - Improved timing accuracy of the model