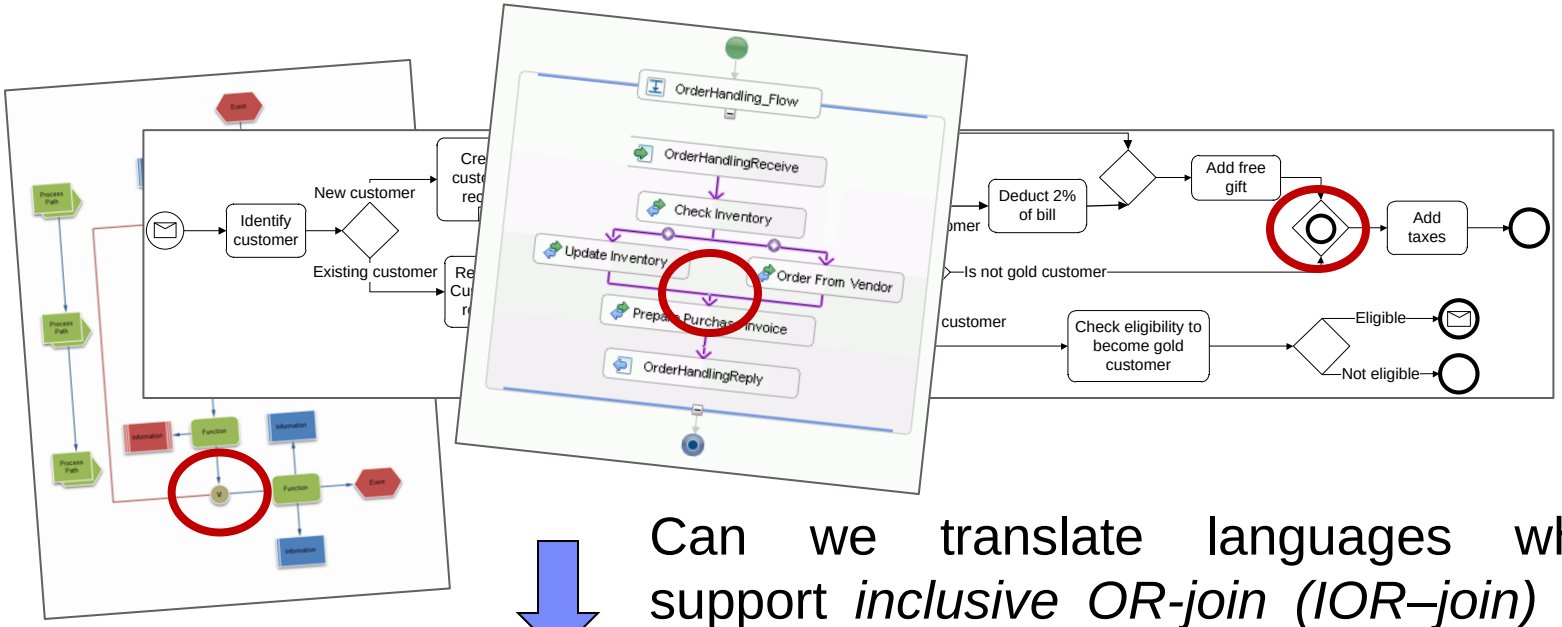
IBM Research – Zurich
Process Management Technologies

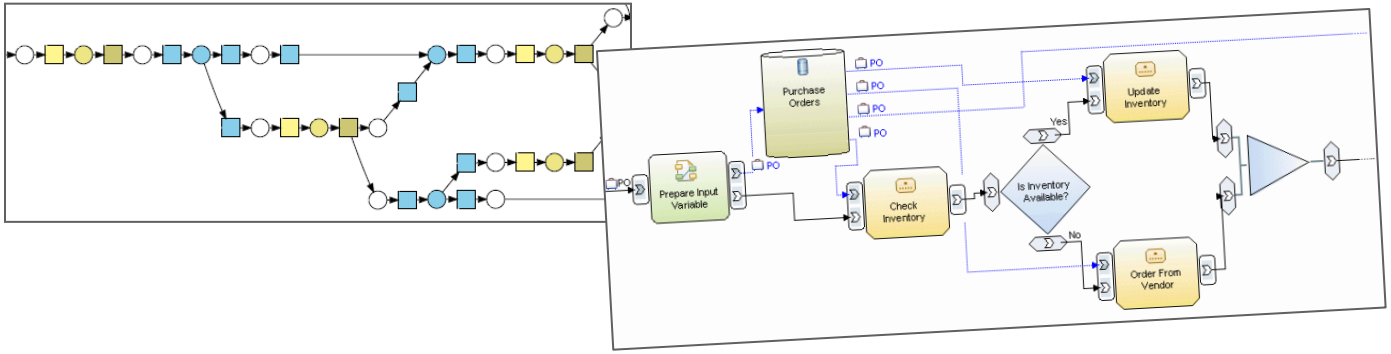
The Difficulty of Replacing an Inclusive OR-Join

Cédric Favre and Hagen Völzer

Translating an inclusive OR-join

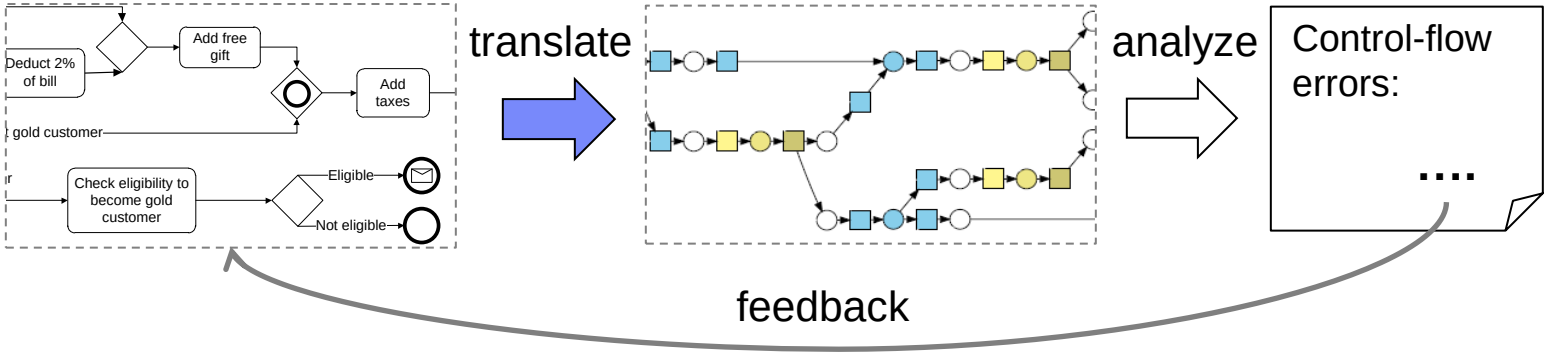


Can we translate languages which support *inclusive OR-join (IOR–join)* into languages which do not ?



Why is it relevant ?

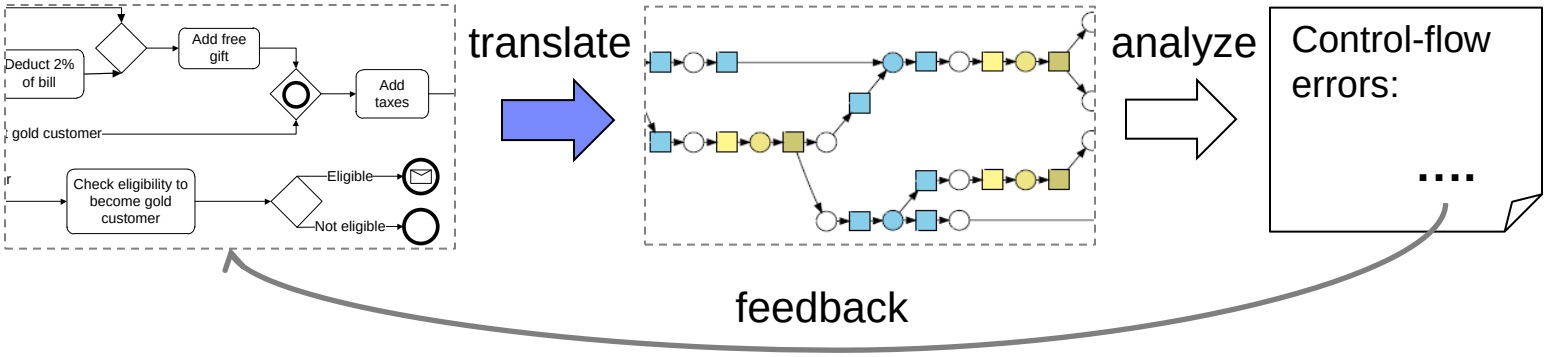
- Many analysis techniques are based on (free-choice) Petri nets: control-flow analysis, cost estimation, performance analysis, process mining …
  - Example: control-flow analysis



- Such translation is necessary to enable the use of Petri net based techniques on industrial business process models containing IOR-joins

Requirements for a translation

- Example: control-flow analysis



1. Equivalence (not difficult on its own)
   - Same “behavior” as original process
2. Size / complexity
   - An exponential blowup of the size of the model is not acceptable
3. Preservation of the original structure
   - The more structure is preserved, the easiest the analysis result can be communicated

Our work

- Restricted to acyclic processes
  - Simpler IOR-join semantics
  - Cycles can be factored out (parsing techniques)
  - IOR-joins are often used in acyclic context (e.g. BPEL)

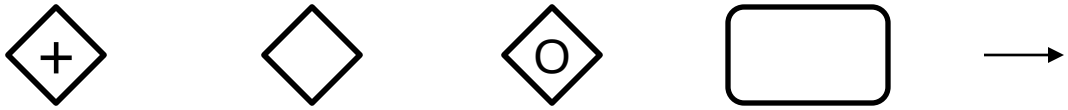
- Translating an IOR-join ?
  - Two replacement techniques and their condition of applicability
    Many occurrences of IOR-joins can be satisfactorily translated
  - An IOR-join that is difficult to translate
    No satisfactory translation for some IOR-joins occurrences

Outline

- Local replacements
- Non-local replacements
- An IOR-join that is difficult to replace

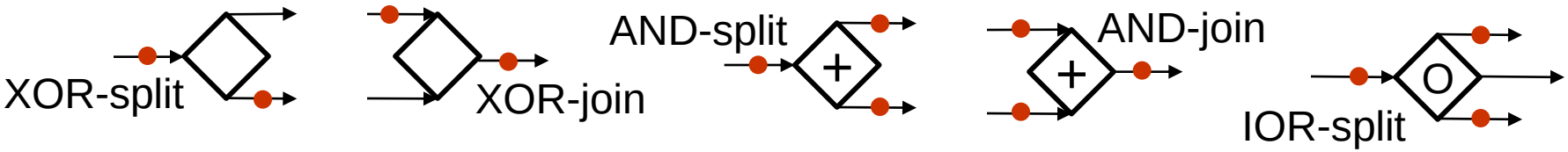
Acyclic workflow graphs

- Parallel, exclusive and inclusive gateways

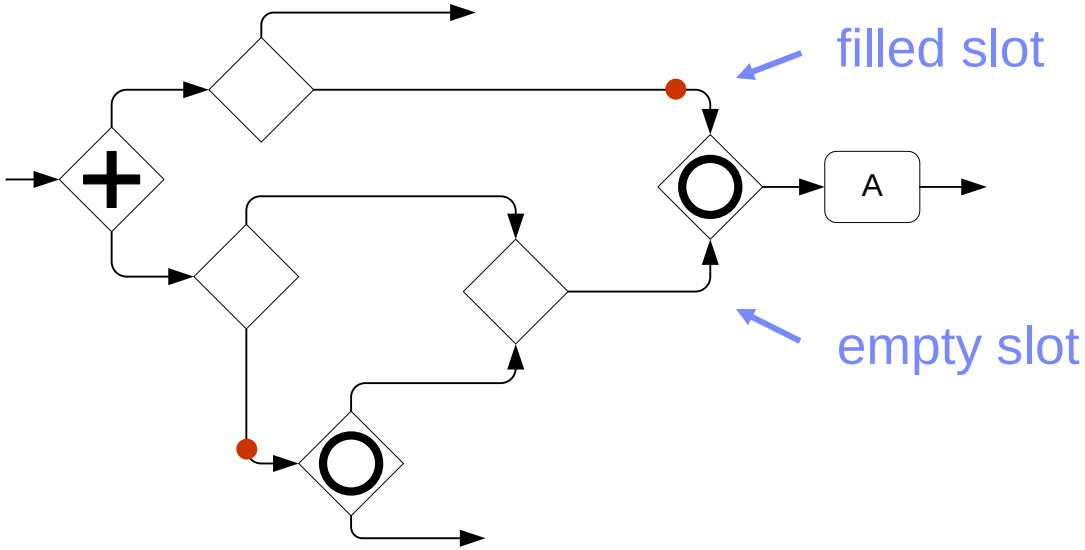


- Workflow graphs without inclusive gateways are isomorphic to free-choice Petri nets
  - **New problem**: Can an acyclic workflow graph containing an IOR-join be transformed into a workflow graph without IOR-joins?

- Semantics:

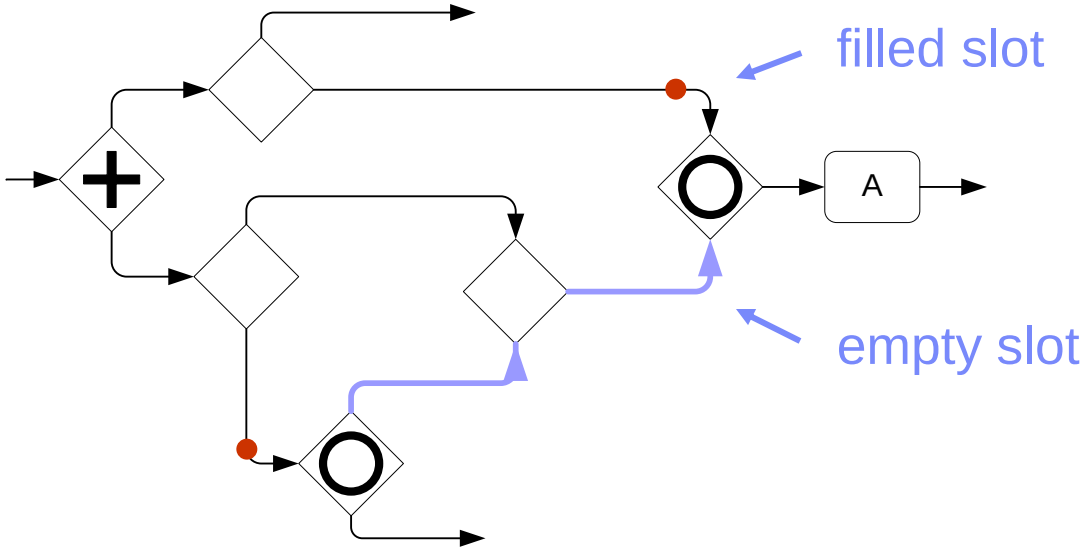


Acyclic IOR-join semantics



- IOR-join is enabled if
  - At least one slot is filled
  - It is not the case that some token "*may still arrive*" on an empty slot

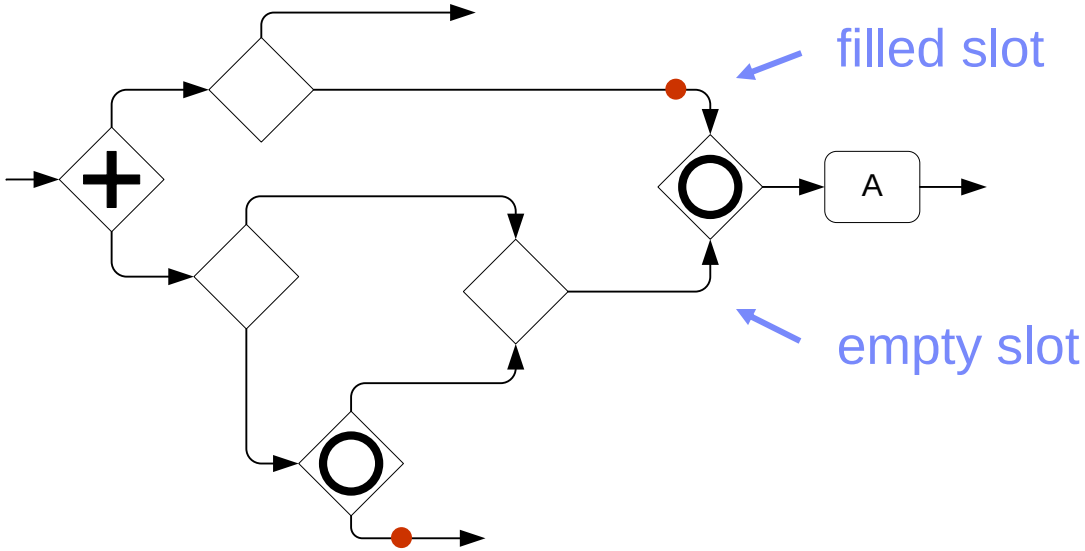
Acyclic IOR-join semantics



- IOR-join is enabled if
  - At least one slot is filled
  - It is not the case that some token "*may still arrive*" on an empty slot

there is a directed path from a token to an empty slot

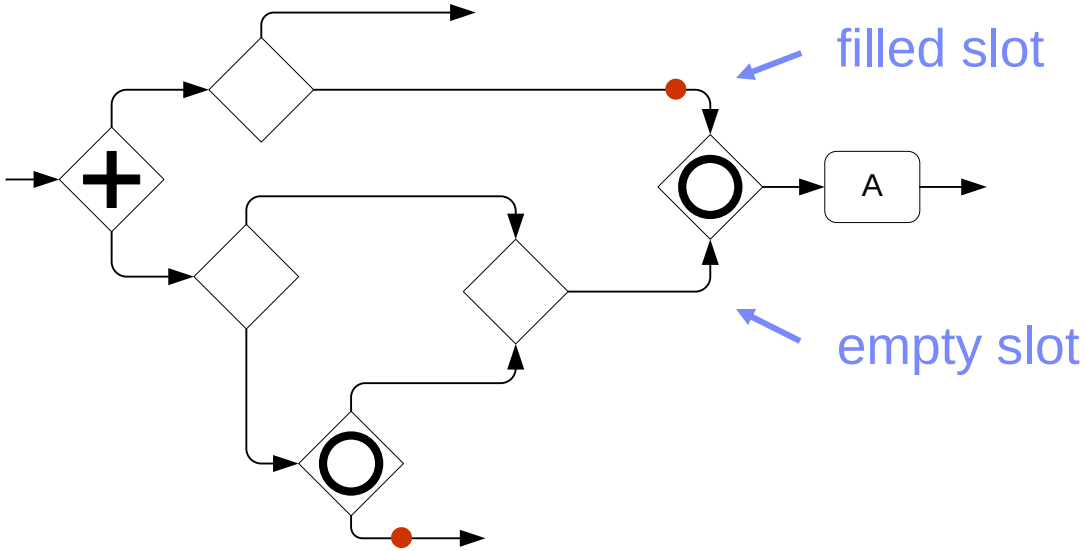
Acyclic IOR-join semantics



- IOR-join is enabled if
  - At least one slot is filled
  - It is not the case that some token "*may still arrive*" on an empty slot

there is a directed path from a token to an empty slot

Acyclic IOR-join semantics



- IOR-join is enabled if
  - At least one slot is filled
  - It is not the case that some token "*may still arrive*" on an empty slot

there is a directed path from a token to an empty slot

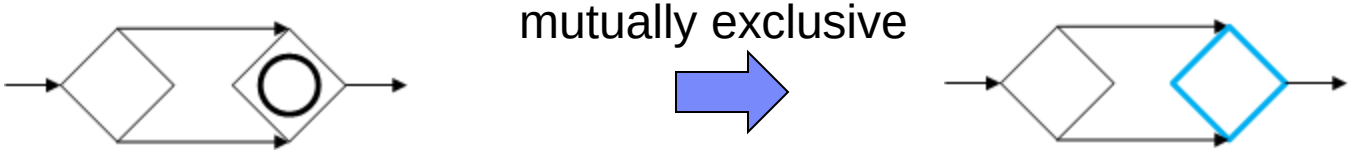
- The semantics is *non-local*
  - Enabling depends on tokens that are not at the join

Outline

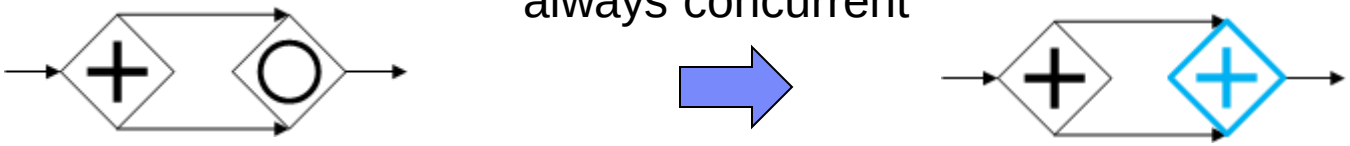
- Local replacements
- Non-local replacements
- An IOR-join that is difficult to replace

A few intuitive replacements

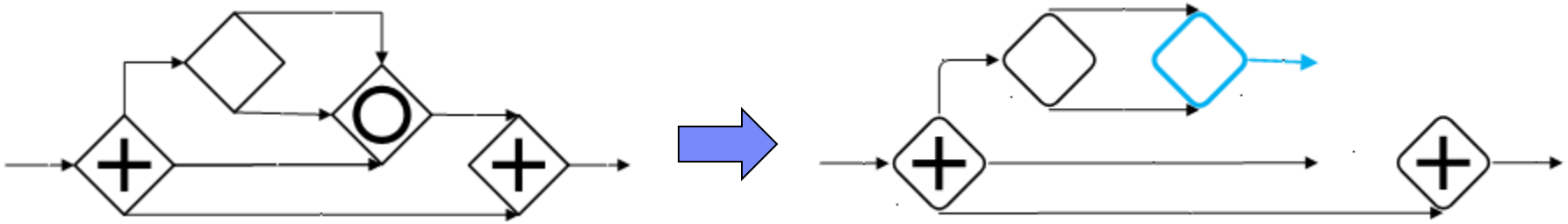
Both edges are mutually exclusive



Both edges are always concurrent

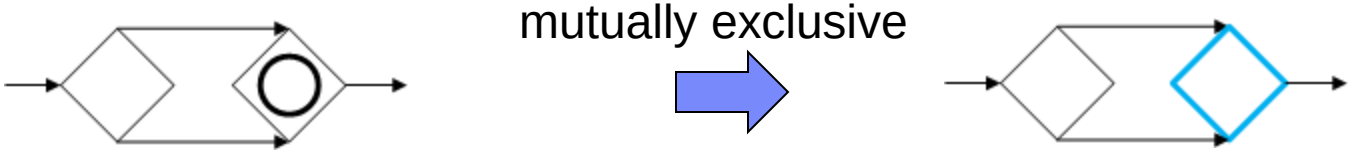


The first two edges are mutually exclusive

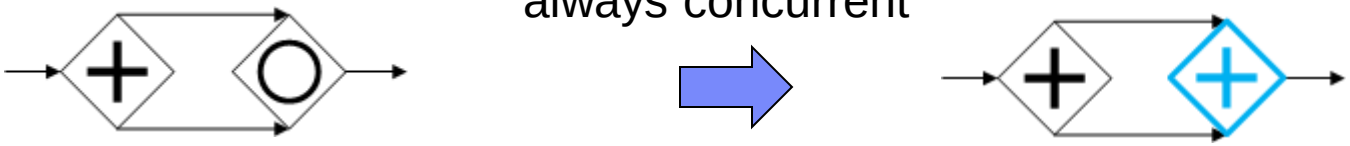


A few intuitive replacements

Both edges are mutually exclusive

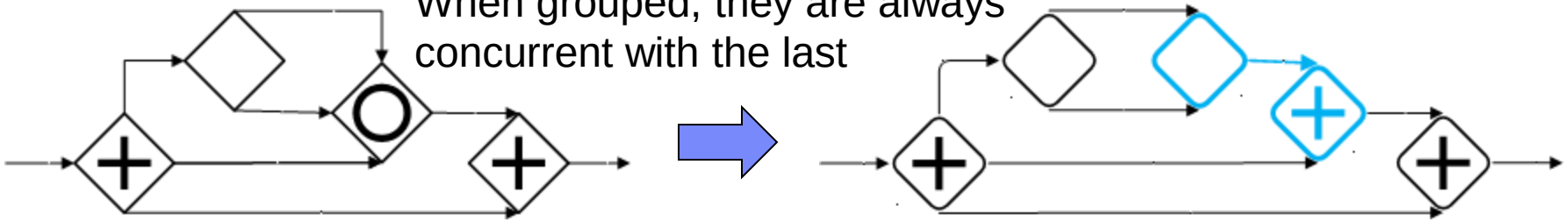


Both edges are always concurrent

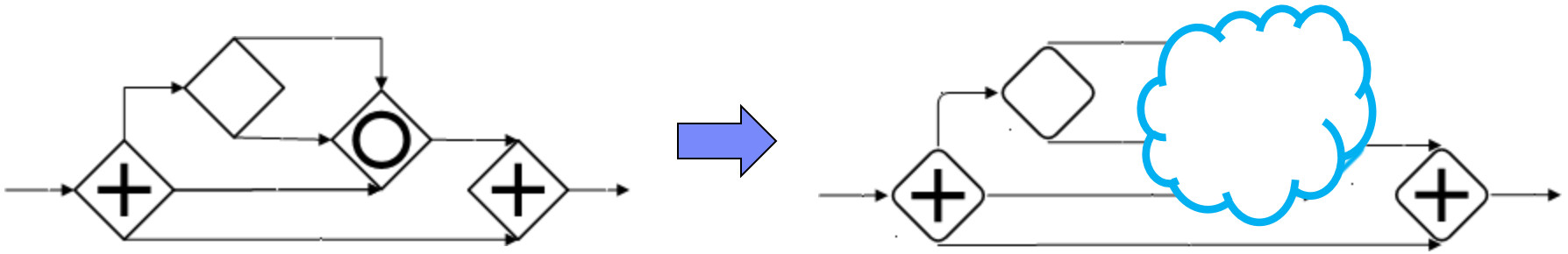


The first two edges are mutually exclusive

When grouped, they are always concurrent with the last



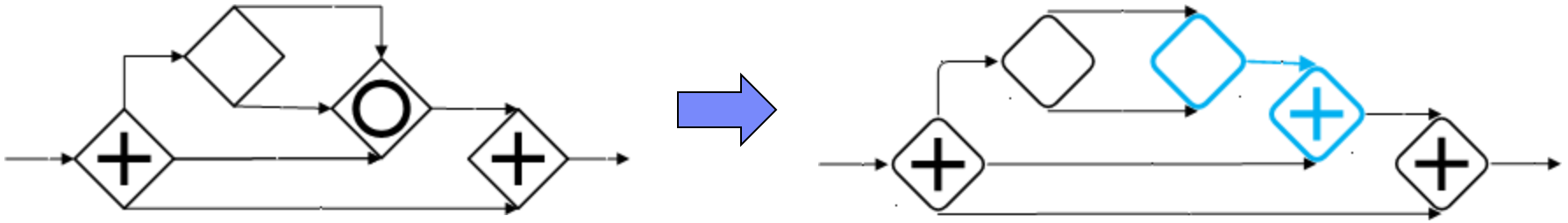
Local replacement



- Replacing IOR-join by a sub-graph (only XOR and AND)
  - Keeping the same set of I/O edges

- Advantages:
  - Intuitive notion of equivalence: “replacement must mimic IOR-join”
  - Structure preserved: rest of the workflow graph unchanged

Local replacement



- Replacing IOR-join by a sub-graph (only XOR and AND)
  - Keeping the same set of I/O edges

- Advantages:
  - Intuitive notion of equivalence: “replacement must mimic IOR-join”
  - Structure preserved: rest of the workflow graph unchanged

Contributions

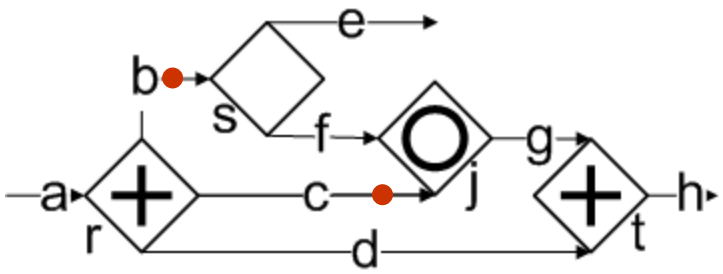
- In the paper
  - Characterization of the IOR-joins that can be replaced locally
  - Replacement technique for all locally replaceable IOR-joins

- Locally replaceable IOR-joins can be identified and replaced

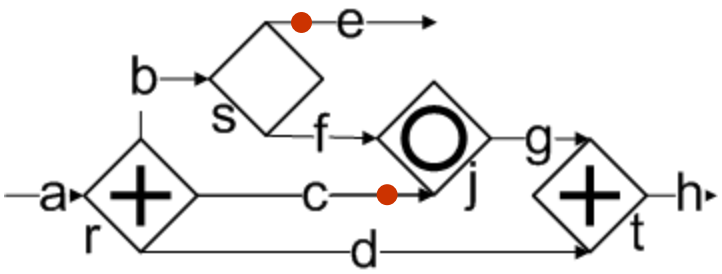
Local replacement – the limit

- The local replacement is using gateways with local semantics and only using local information

- The local information is not enough to differentiate:



Not enabled

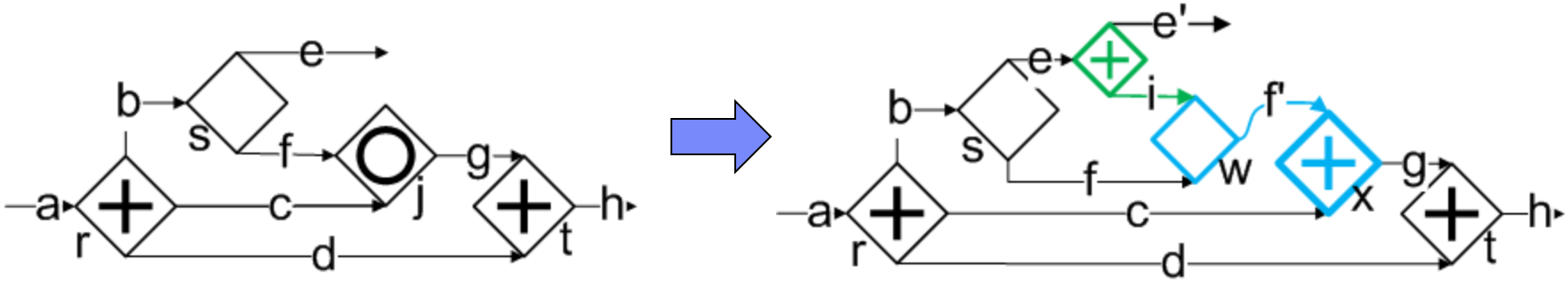


Enabled

Outline

- Local replacements
- Non-local replacements
- An IOR-join that is difficult to replace

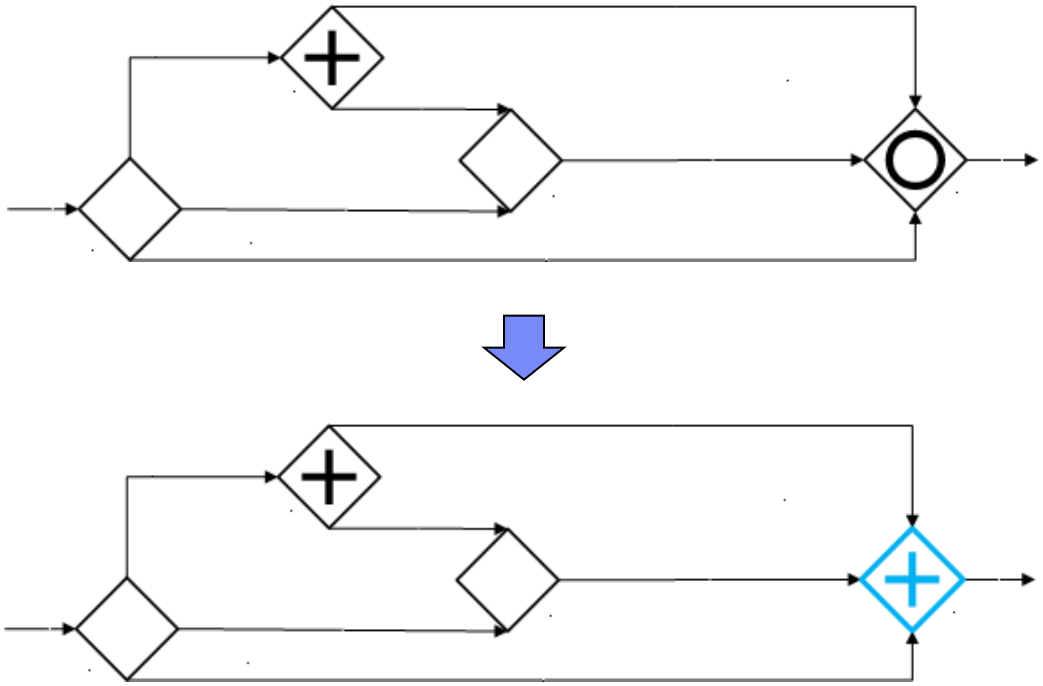
Non-local replacements



- Idea
  - Transmit the missing information using bridges
- Non-local replacement = local replacement + bridges

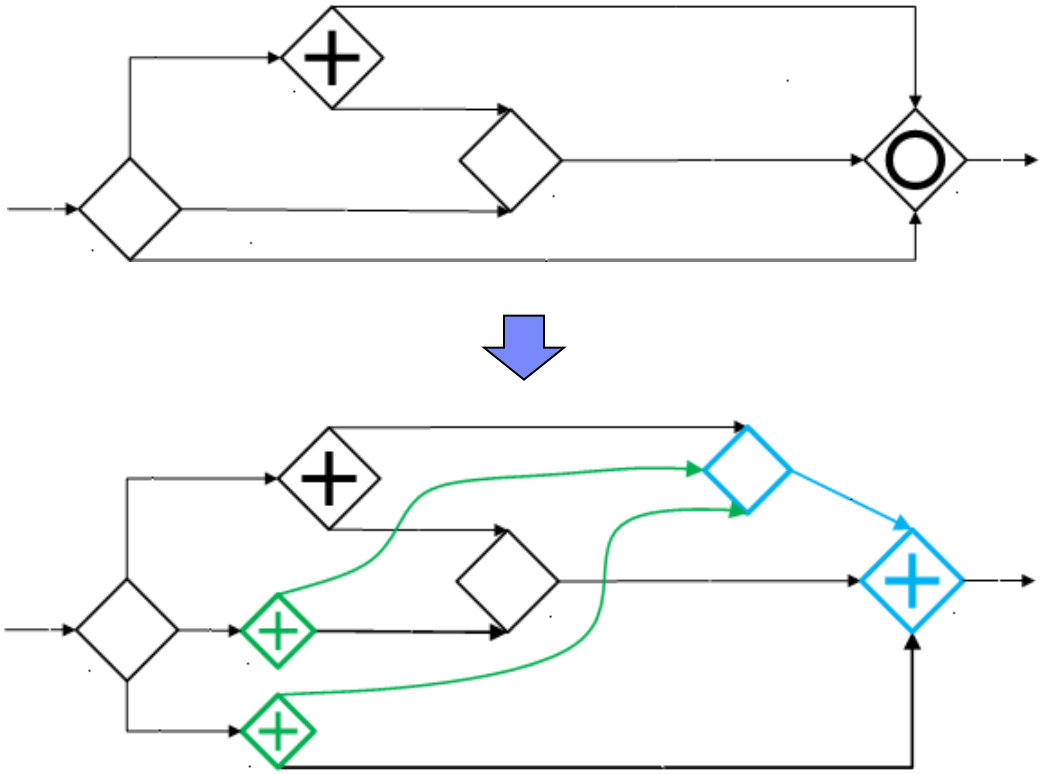
Simple K-replacement

- Idea [Kiepuszewski et al., completion of processes]
  - Replace IOR-join by an AND-join
  - For each incoming edge, “bridge” all the tokens that are “lost”



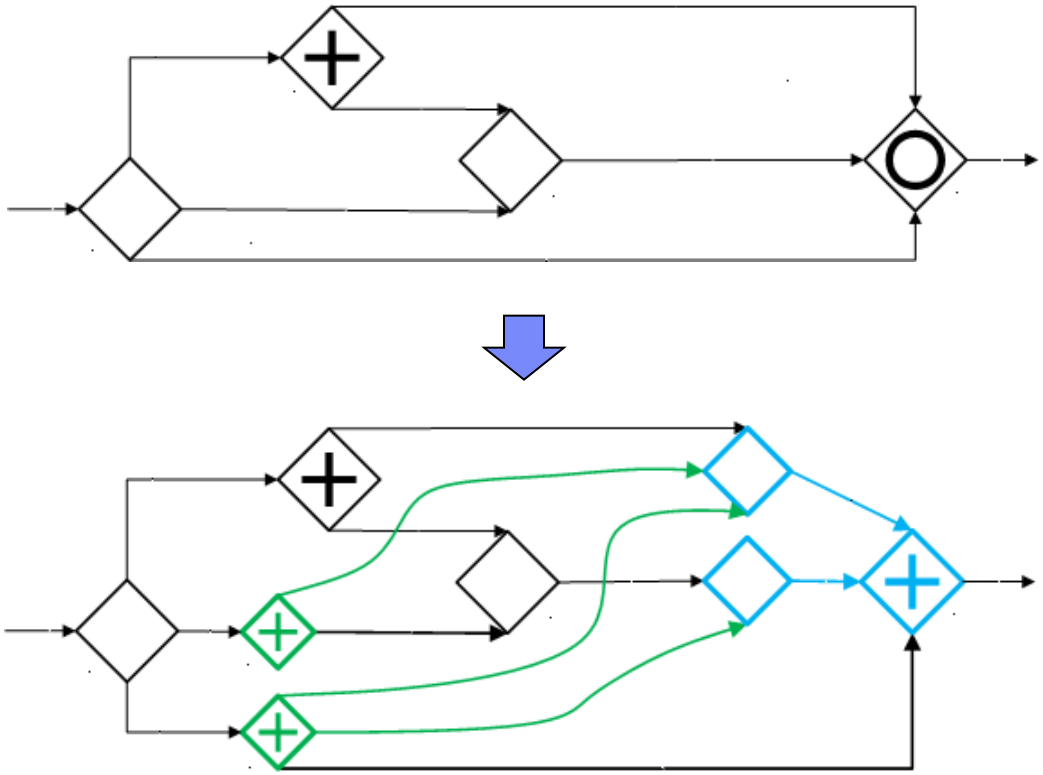
Simple K-replacement

- Idea [Kiepuszewski et al., completion of processes]
  - Replace IOR-join by an AND-join
  - For each incoming edge, “bridge” all the tokens that are “lost”



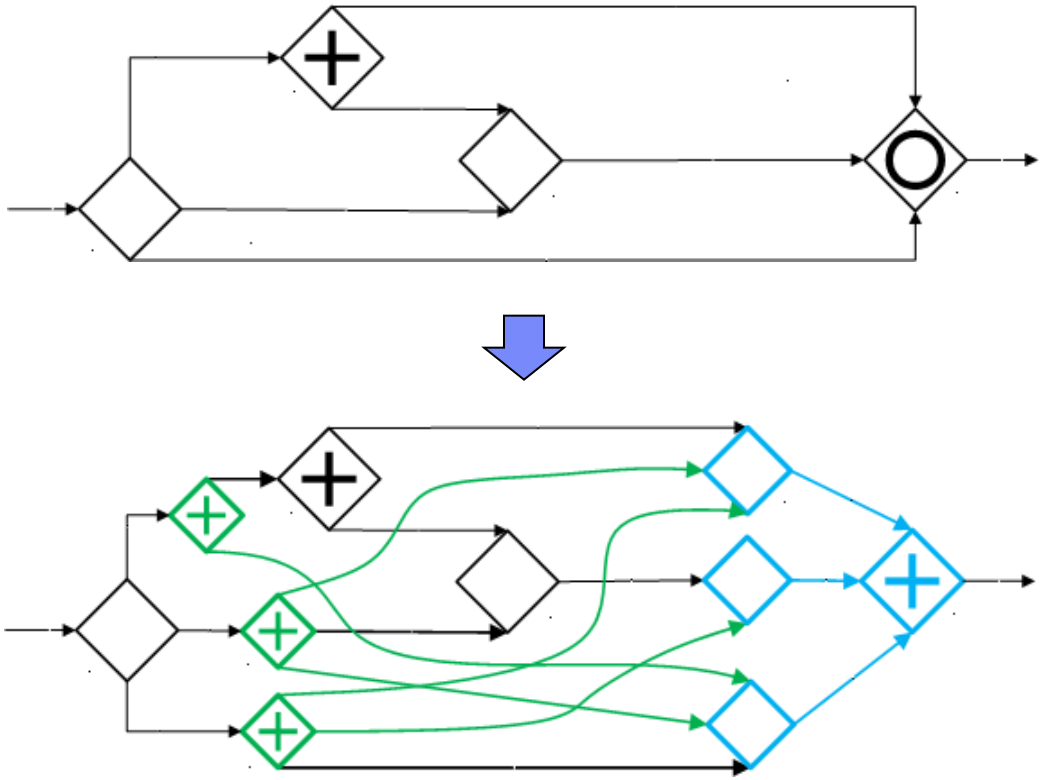
Simple K-replacement

- Idea [Kiepuszewski et al., completion of processes]
  - Replace IOR-join by an AND-join
  - For each incoming edge, “bridge” all the tokens that are “lost”



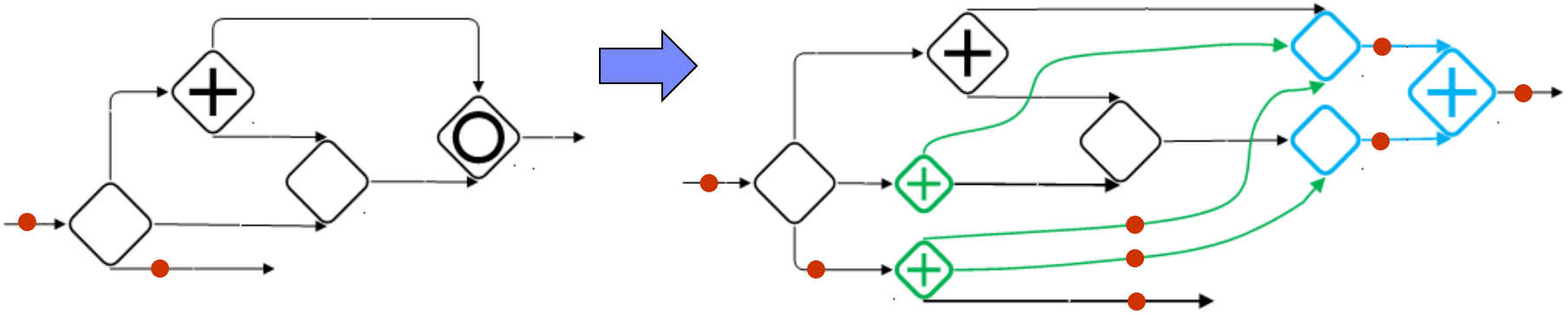
Simple K-replacement

- Idea [Kiepuszewski et al., completion of processes]
  - Replace IOR-join by an AND-join
  - For each incoming edge, “bridge” all the tokens that are “lost”



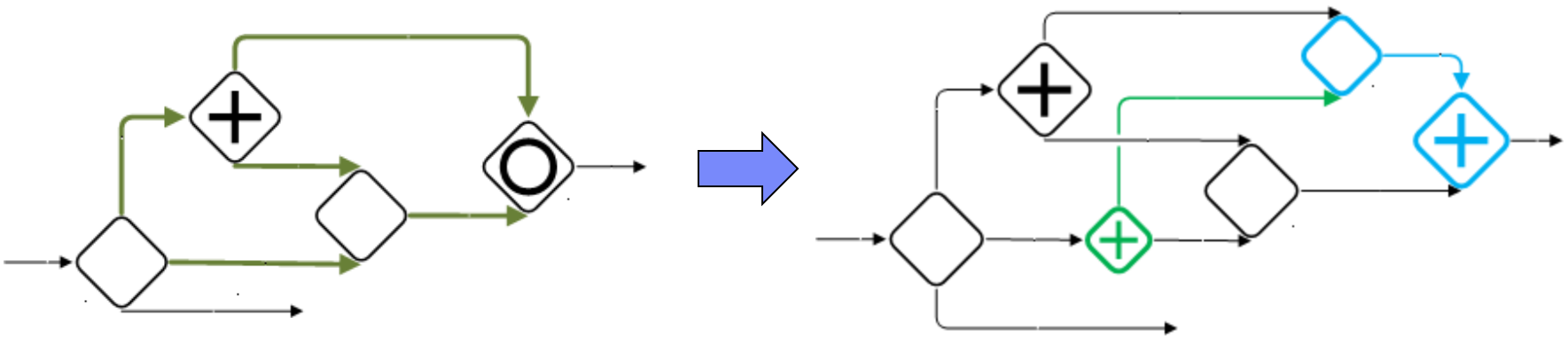
Simple K-replacement problem

- The replacement is executed during every execution, but some IOR-joins do not



- Idea
  - Reduce the area of application

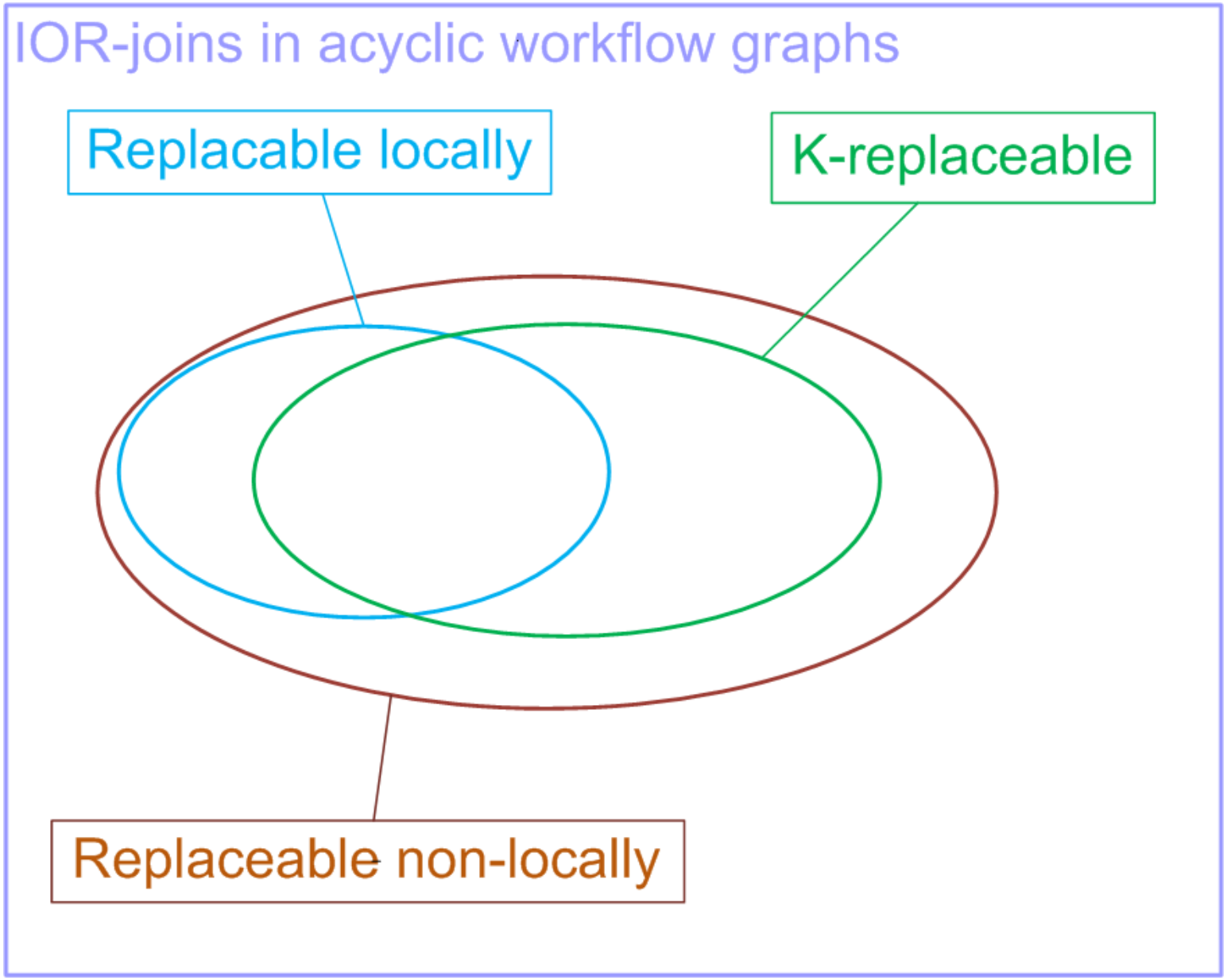
K-replacement - reduced area of application



- In the paper
  - ***K-replacement*** (reduced area of application)
  - A condition of applicability which can be checked in polynomial time

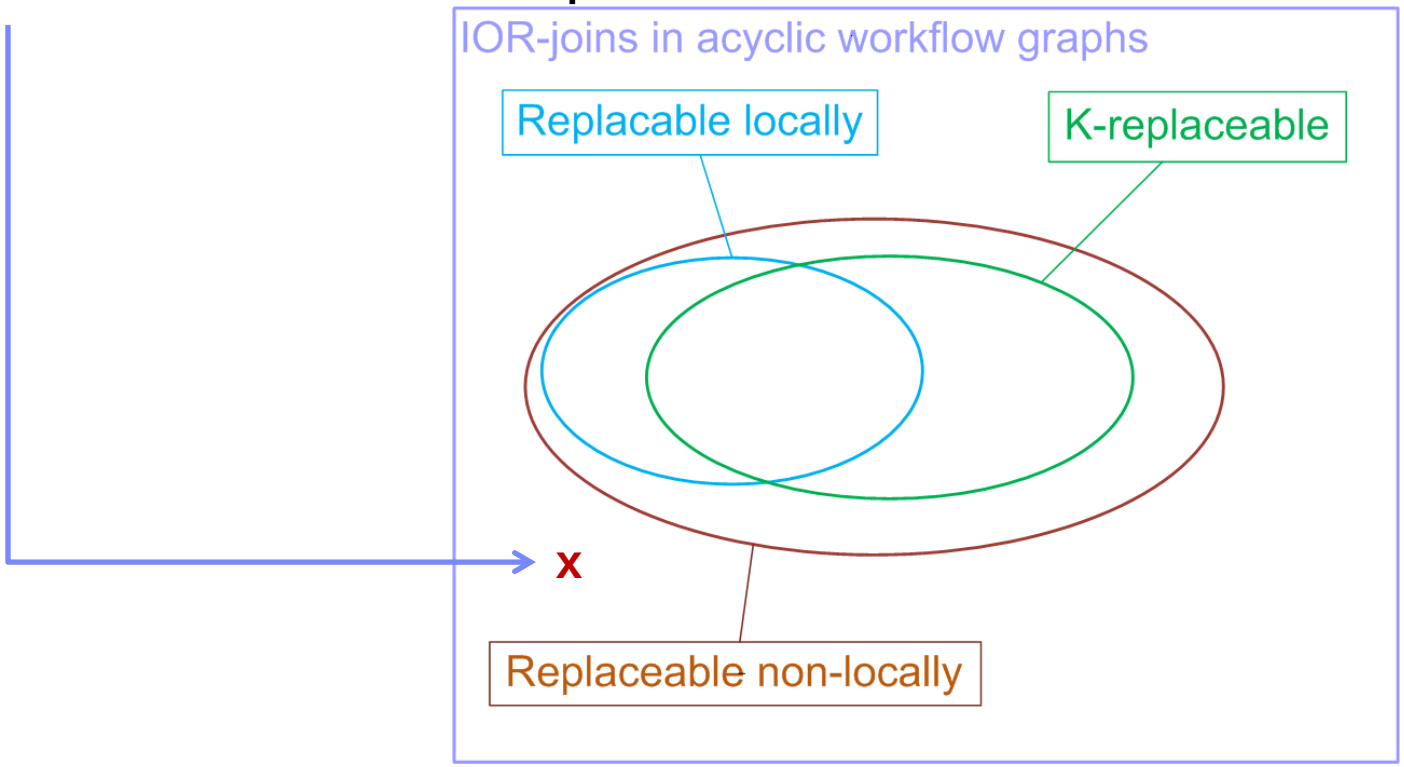
- We can identify K-replaceable IOR-joins and replace them in polynomial time

IOR-joins in acyclic workflow graphs

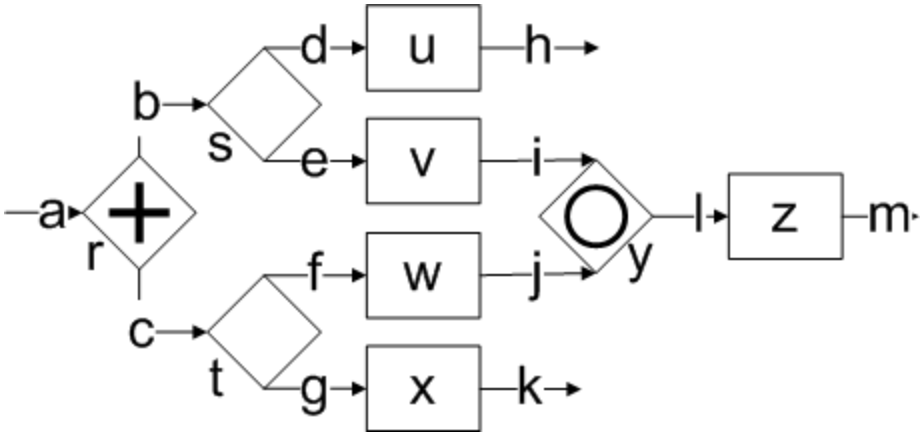


Outline

- Local replacements
- Non-local replacements
- An IOR-join that is difficult to replace

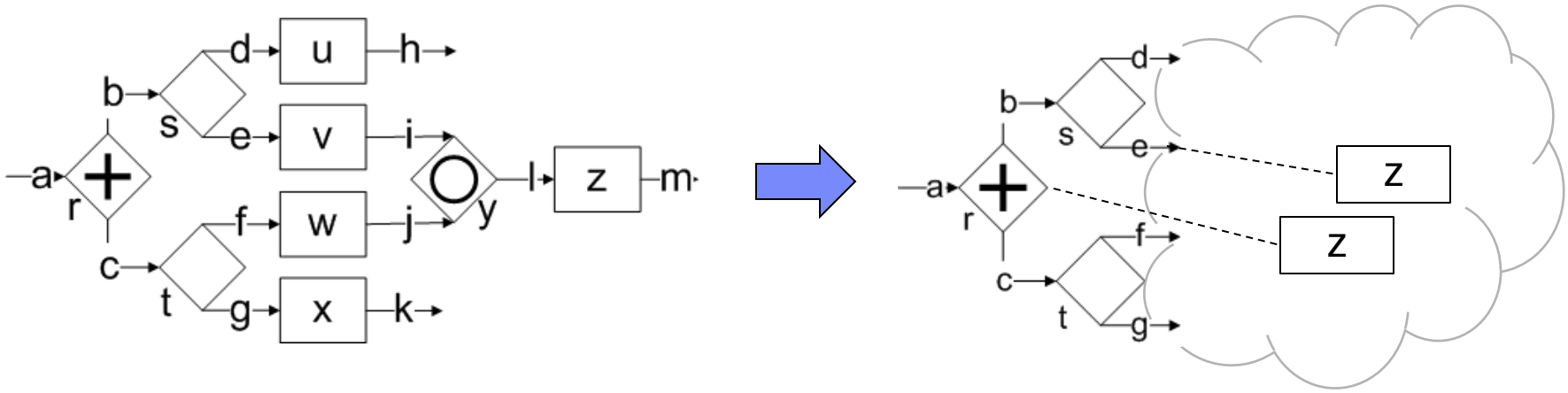


An IOR-join which is difficult to replace



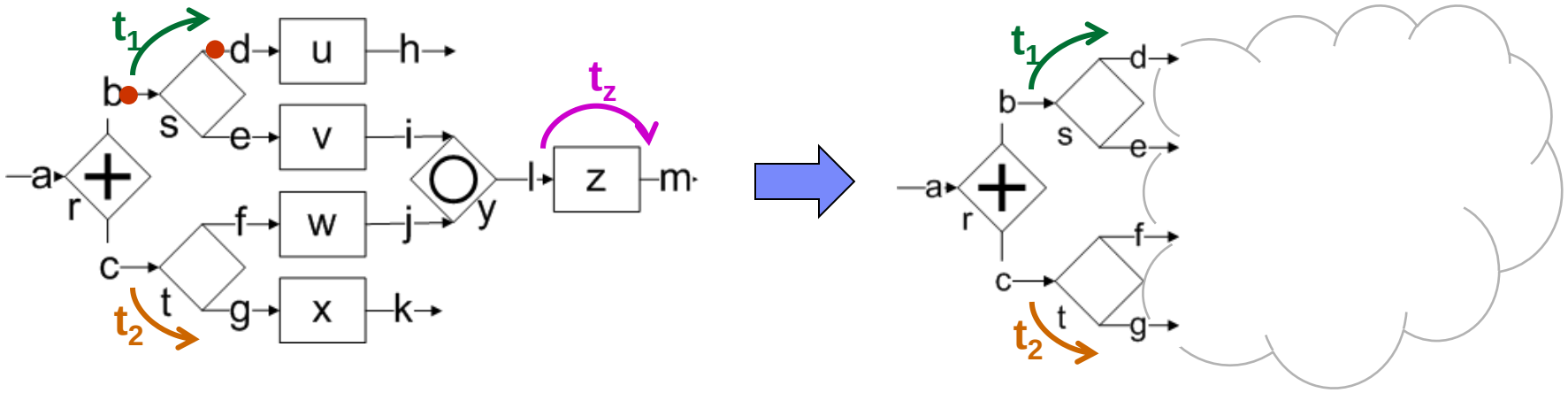
- Cannot be:
  - locally replaced,
  - K-replaced
- We will show that the synchronization role played by *y* cannot be obtained using AND and XOR logic

The synchronization role of y cannot be replaced



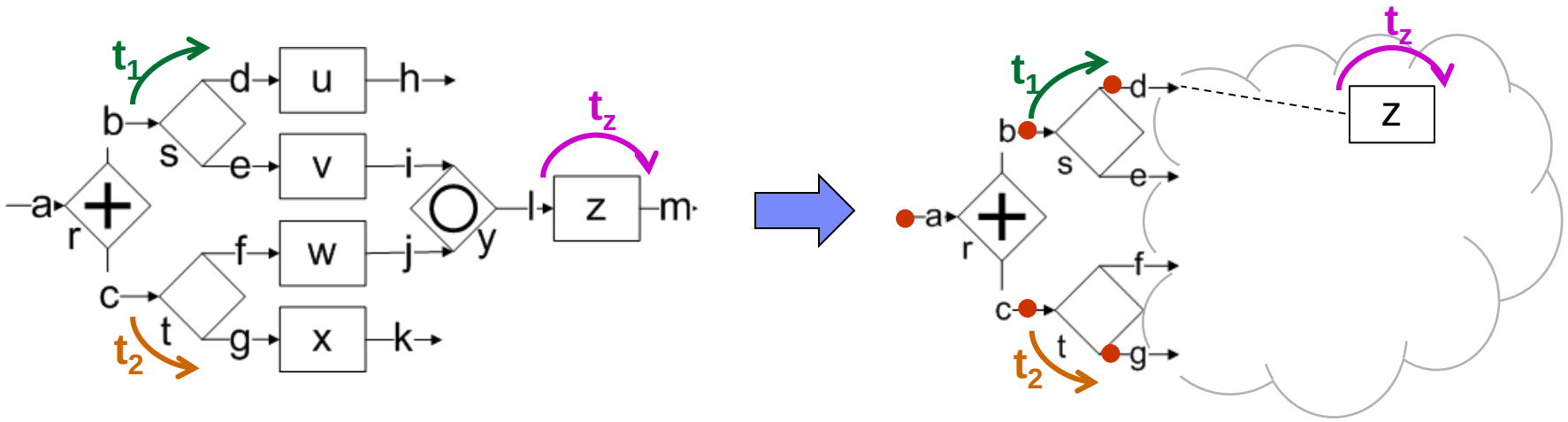
- Question: can we complete the prefix to obtain same behavior
  - Tasks can be copied, edges of the prefix can be bridged
- In the paper, we show that the prefix cannot be completed using AND and XOR logic

The prefix cannot be completed



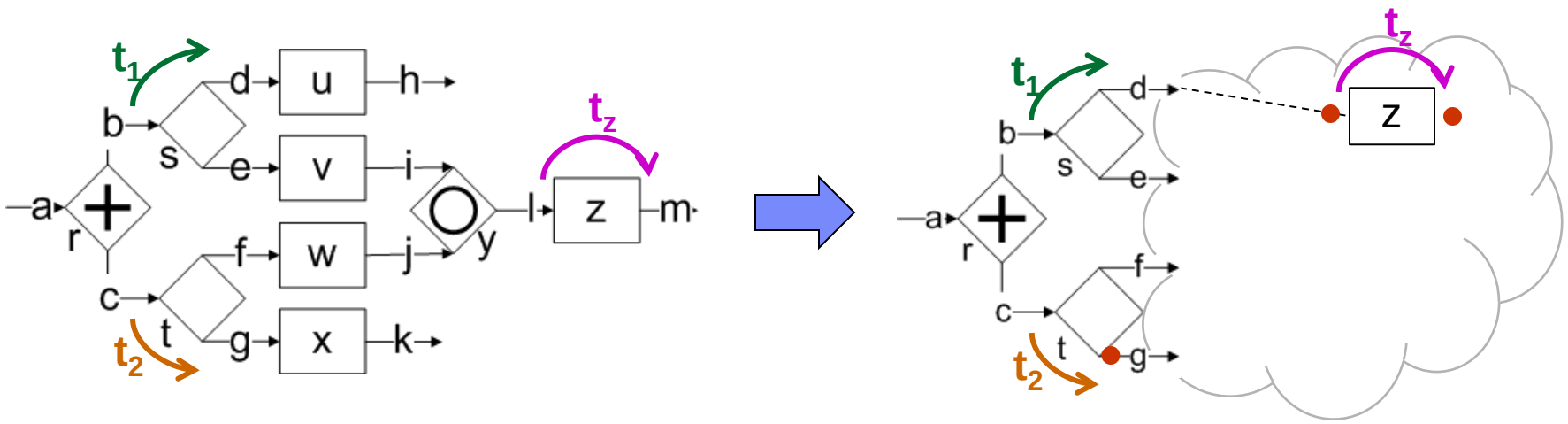
- Execution properties
  - There exist executions where t_1 and t_z execute and, in executions where t_1 and t_z are executed, t_1 always executes before t_z
  - There is no execution executing t_1, t_2, and t_z

The prefix cannot be completed



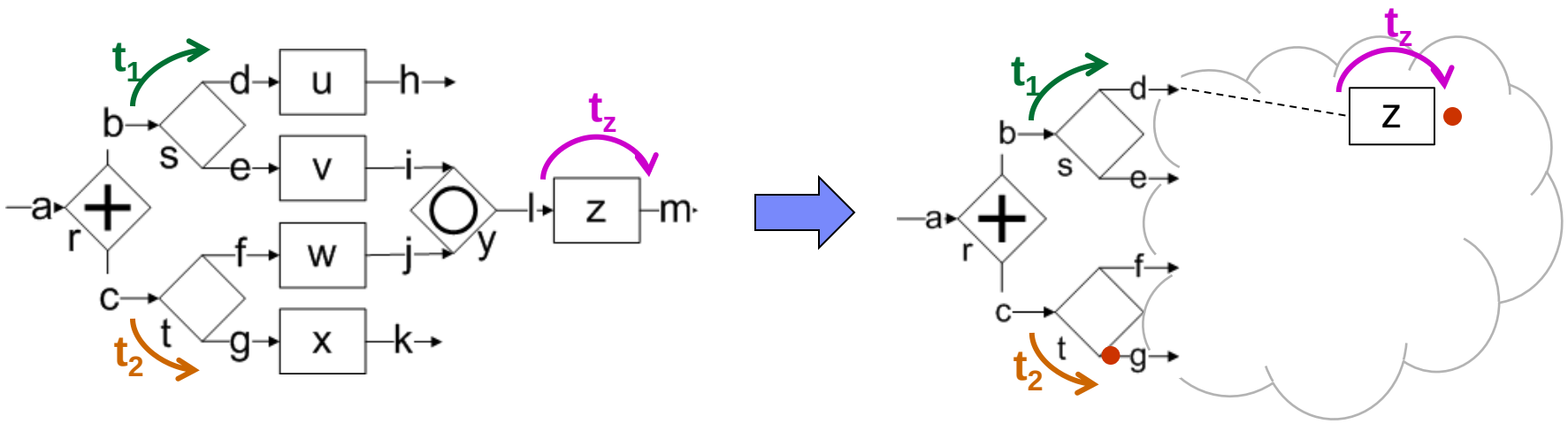
- Execution properties
  - There exist executions where t_1 and t_z execute and, in executions where t_1 and t_z are executed, t_1 always executes before t_z
    → **there is a path from t_1 to t_z in the completion**
  - There is no execution executing t_1, t_2, and t_z

The prefix cannot be completed



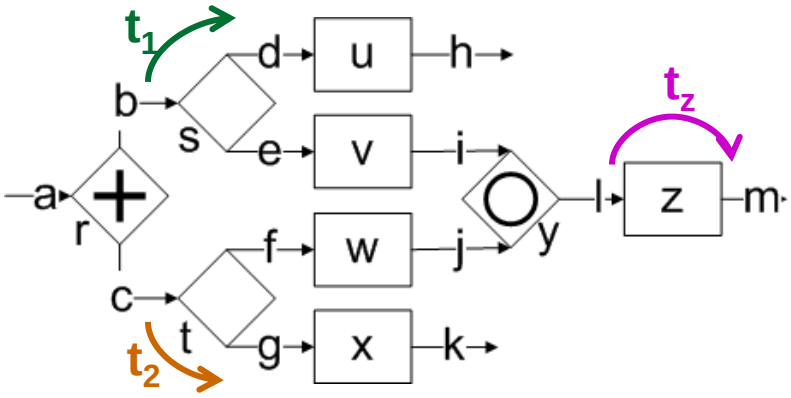
- Execution properties
  - There exist executions where t_1 and t_z execute and, in executions where t_1 and t_z are executed, t_1 always executes before t_z
    → **there is a path from t_1 to t_z in the completion**
    → **this path can be executed**
  - There is no execution executing t_1, t_2, and t_z

The prefix cannot be completed



- Execution properties
  - There exist executions where t_1 and t_z execute and, in executions where t_1 and t_z are executed, t_1 always executes before t_z
    → **there is a path from t_1 to t_z in the completion**
    → **this path can be executed**
  - **There is no execution executing t_1, t_2, and t_z**

Conclusions from this example



- For any notion of equivalence which implies the conservation of the execution properties
  - There exist executions where t_1 and t_z execute and, in executions where t_1 and t_z are executed, t_1 always executes before t_z
  - There is no execution executing t_1, t_2, and t_z

- The prefix cannot be completed:
  - The synchronization role played by an IOR-join cannot, in general, be implemented using XOR and AND gateways

Summary

- Contributions:
  - For acyclic processes
    - Characterization of IOR-joins that can be replaced locally and a local replacement technique
    - A non-local replacement technique with its condition of applicability which run in polynomial time
  - Show that a simple processes contains an IOR-join which synchronization role cannot be replaced by XOR and AND logic

- For some use cases, we have to be careful when we want to apply Petri net based analysis techniques to models that might contain an IOR-join