Well-Structured Graph Transformation Systems

Barbara König

Universität Duisburg-Essen, Germany

Joint work with Jan Stückrath and Salil Joshi

Context

Our current research on verification of graph transformation systems:

- Graph specification languages and graph automata
  (Christoph Blume & Dennis Nolte & Sebastian Küpper)
- Termination of graph transformation systems
  (Sander Bruggink & Hans Zantema, Eindhoven)
- Backward analysis for well-structured graph transformation systems
  (Jan Stückrath)

Motivation

Our aim in this talk

Given a graph transformation system with an initial graph G_0, find a procedure for verifying whether a given graph G can be "covered", starting from G_0.

Our toolbox

- Well-structured transition systems
  the state-of-the-art method for obtaining decidability results for infinite-state systems
- Graph theory
  especially: graph minor theory and well-quasi orders on graphs
- Graph transformation theory
  SPO, pushouts, ...

[CAV '08] [CONCUR '14]

Overview

Graph Minor Theory

- Graph minor theory by Robertson and Seymour
- Long series of papers (Graph minors I–XXIII)
- Deep graph-theoretical results with applications in computer science (mainly efficient algorithms, complexity theory)
- What about applications in verification?

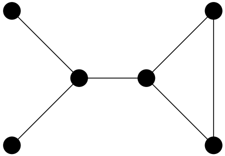
Graph Minor Theory

Minor of a graph

The minors of a graph G can be obtained by (iteratively)

- Deleting edges.
- Deleting isolated nodes.
- Contracting edges.

We write $M \leq G$ if M is a minor of G.



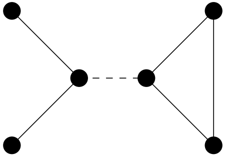
Graph Minor Theory

Minor of a graph

The minors of a graph G can be obtained by (iteratively)

- Deleting edges.
- Deleting isolated nodes.
- Contracting edges.

We write $M \leq G$ if M is a minor of G.



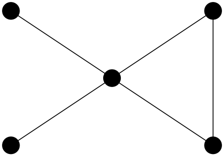
Graph Minor Theory

Minor of a graph

The minors of a graph G can be obtained by (iteratively)

- Deleting edges.
- Deleting isolated nodes.
- Contracting edges.

We write $M \leq G$ if M is a minor of G.



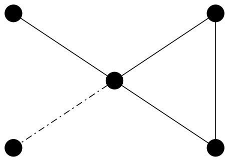
Graph Minor Theory

Minor of a graph

The minors of a graph G can be obtained by (iteratively)

- Deleting edges.
- Deleting isolated nodes.
- Contracting edges.

We write $M \leq G$ if M is a minor of G.



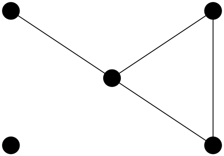
Graph Minor Theory

Minor of a graph

The minors of a graph G can be obtained by (iteratively)

- Deleting edges.
- Deleting isolated nodes.
- Contracting edges.

We write $M \leq G$ if M is a minor of G.



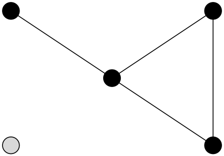
Graph Minor Theory

Minor of a graph

The minors of a graph G can be obtained by (iteratively)

- Deleting edges.
- Deleting isolated nodes.
- Contracting edges.

We write $M \leq G$ if M is a minor of G.



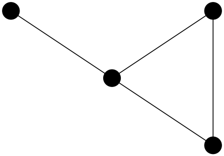
Graph Minor Theory

Minor of a graph

The minors of a graph G can be obtained by (iteratively)

- Deleting edges.
- Deleting isolated nodes.
- Contracting edges.

We write $M \leq G$ if M is a minor of G.



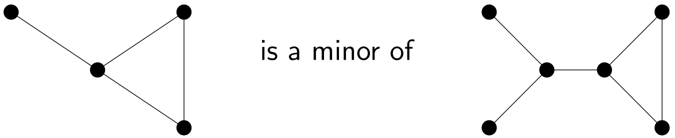
Graph Minor Theory

Minor of a graph

The minors of a graph G can be obtained by (iteratively)

- Deleting edges.
- Deleting isolated nodes.
- Contracting edges.

We write $M \leq G$ if M is a minor of G.



Graph Minor Theory

Graph minor theorem (Robertson & Seymour)

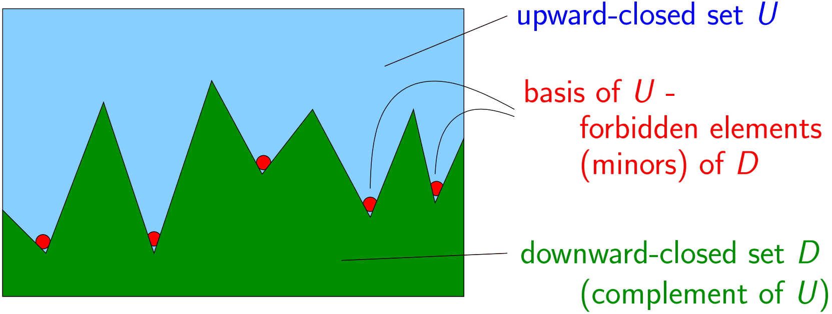
In every infinite sequence $G_0, G_1, G_2, G_3, \ldots$ there exist indices $i < j$ such that G_i is a minor of G_j.

In other words: the minor ordering $\leq$ is a well-quasi-order (wqo).

Graph Minor Theory

Consequences:

- every upward-closed set of graphs has a finite basis (i.e., a finite set of minimal elements)
- every downward-closed set of graphs can be characterized by finitely many forbidden minors.



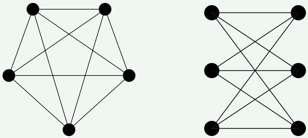
Graph Minor Theory

Downward-closed sets of graphs:

- Graphs that are disjoint unions of paths
- Forests
- Planar graphs
- Graphs that can be embedded in a torus
- ...

Kuratowski's theorem

A graph is planar if and only if it does not contain the K_5 and the $K_{3,3}$ as a minor.



Graph Minor Theory

What about labelled graphs, directed graphs, hypergraphs?

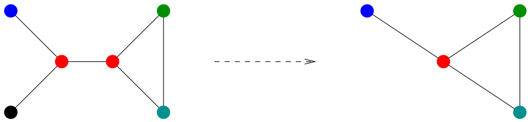
$\rightsquigarrow$ The graph minor theorem holds even for labelled hypergraphs!
(If an edge is contracted, its incident nodes are arbitrarily partitioned and merged.)

Minor morphisms

$H \leq G$ iff there exists a minor morphism $G \mapsto H$, that is

- there is a partial graph morphism $G \rightharpoonup H$,
- which is surjective, injective on edges *and*
- whenever two nodes v, w of G are mapped to z in H, there exists an (undirected) path between v, w which is contracted.

A minor morphism:



Graph Minor Theory

There are other interesting well-quasi-orders on restricted sets Q graphs, for instance:

set of graphs Q	well-quasi-order
all graphs	minor ordering
graphs with a bound on the longest undirected path	subgraph ordering
graph with a bound on the longest undirected path and on the number of parallel edges	induced subgraph ordering

All these orders can be characterized by a class of order morphisms (analogously to minor morphisms), symbolically: $\mapsto$

Well-Structured Transition Systems

Well-quasi-orders are also an important ingredient of well-structured transition systems (WSTS) [Finkel/Schnoebelen, Abdulla et al.]

WSTS (Well-structured transition system)

Let S be a set of states, $\Rightarrow$ a transition relation and $\leq$ a partial order on states. The transition system is well-structured if

- $\leq$ is a well-quasi-order
- Whenever $s_1 \leq t_1$ and $s_1 \Rightarrow s_2$, there exists a state t_2 such that $t_1 \Rightarrow^* t_2$ and $s_2 \leq t_2$ (compatibility condition).

$$\begin{array}{ccc} t_1 & \Longrightarrow^* & t_2 \\ \vee\!| & & \vee\!| \\ s_1 & \Longrightarrow & s_2 \end{array}$$

Well-Structured Transition Systems

The prototypical example for a WSTS are Petri nets:

- States: markings
- Transition relation: firing of transitions as specified by the net
- Well-quasi-order: $m_1 \le m_2$ if m_2 covers m_1 (m_2 contains at least as many tokens in every place)

Other examples:

- Context-free string rewrite systems
- Basic process algebra
- "Lossy" systems
- Systems with home-states

Well-Structured Transition Systems

Backward Reachability

Take a set $I \subseteq S$ of states and compute $Pred^*(I)$ (the set of all predecessors) as the limit of the sequence

$$I_0 = I \qquad I_{i+1} = I_i \cup Pred(I_i),$$

where *Pred* returns the direct predecessors of a set of states.

Backward Reachability and WSTS

In the case of WSTS it holds that

- If I is upward-closed (and hence representable by a finite basis), then $Pred^*(I)$ is upward-closed.
- The sequence $I_0, I_1, I_2, \ldots$ eventually becomes stationary, i.e., $\uparrow I_n = \uparrow I_{n+1}$ (upward closures coincide) and $Pred^*(I) = \uparrow I_n$.

Well-Structured Transition Systems

Covering problem

Covering problem: Given an initial state s_0 and another state s_f. Can we reach a state s from s_0, i.e., $s_0 \Rightarrow^* s$ such that $s \geq s_f$?

The covering problem for WSTS is decidable if

- we can effectively compute a finite basis for (the upward-closure of) $Pred(I)$ whenever we have a finite basis for I *and*
- if the well-quasi order $\leq$ is decidable.

Procedure: Compute $Pred^*(\uparrow\{s_f\})$ and check whether it contains s_0.

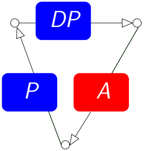
Graph Transformation Systems

Question: can we view (some) graph transformation systems (single-pushout approach) as well-structured transition systems?

Running example: Termination detection

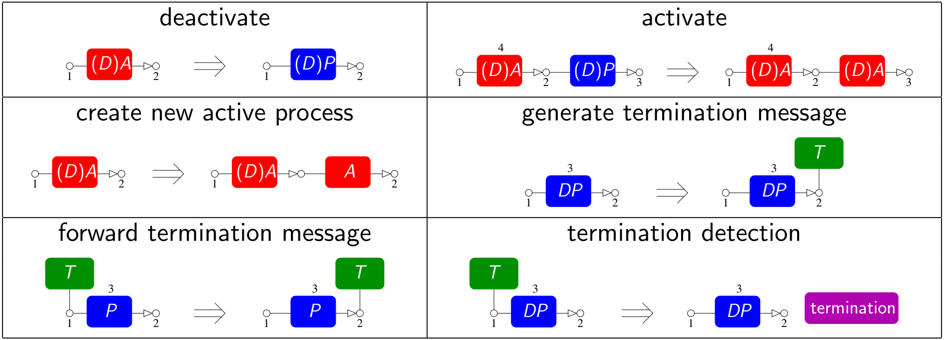
- A ring consisting of active and passive processes.

Start graph:



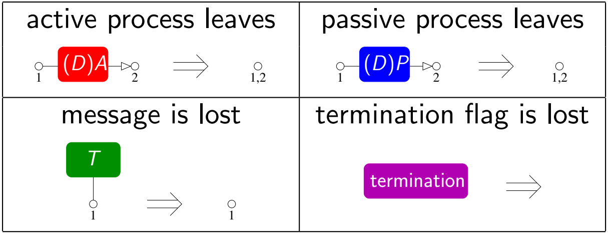
- Active processes may become passive at any time.
- Active processes may activate passive processes and create new active processes.
- There is a special process (the detector *DA*, *DP*) that may generate a message for termination detection.
- This message is forwarded by passive processes and received by the (passive) detector which then declares termination.

Running example: Termination detection



Running example: Termination detection

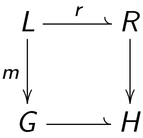
Additionally: The system is unreliable. Processes may leave the ring at any time and messages may get lost.

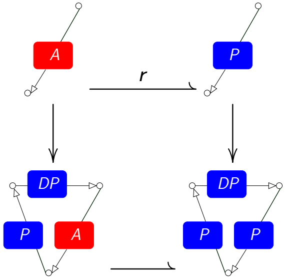


SPO (single pushout) rewriting rules, given by partial graph morphisms from the left-hand side to the right-hand side.

Single-pushout approach

Take the pushout of the partial rule morphism ($r\colon L \rightharpoonup R$) and the total match ($m\colon L \to G$) in the category of partial graph morphisms in order to obtain the resulting graph H.





Construct H by

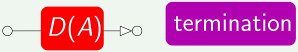
- deleting elements of G which are undefined under r
- creating elements which are new in R

It can be shown that our order morphisms (minor morphisms, etc.) are preserved by pushouts along total morphisms (important for our theory!)

Running example: Termination detection

Correctness

- Are the rules incorrect?
- That is, can we reach a graph where termination has been declared, but there are still active processes?
- Can we reach a graph which contains the following graph as a minor?



$\rightsquigarrow$ View graph transformation as a WSTS (with the minor ordering) and solve the covering problem for the graph above via backward analysis!

GTS as Well-Structured Transition Systems

Graph transformation systems are in general Turing-complete $\rightsquigarrow$ not all GTS can be well-structured

But some subclasses are WSTS with respect to the minor ordering:

- Context-free graph grammars
- GTS where the left-hand sides consist of disconnected edges
- GTS which contain edge contraction rules for every edge label ("lossy" systems)

GTS as Well-Structured Transition Systems

Obtaining a WSTS by adding edge contraction rules

$$\begin{array}{ccc} H_1 & & \\ \begin{rotate}{90}\leq\end{rotate} & & \\ G_1 & \stackrel{r}{\Longrightarrow} & G_2 \end{array}$$

If G_1 is a minor of H_1 and G_1 is rewritten to G_2 ...

GTS as Well-Structured Transition Systems

Obtaining a WSTS by adding edge contraction rules

$$\begin{array}{ccc} H_1 & \Longrightarrow^{*} H' & \\ \vee\!| & & \\ G_1 & \stackrel{r}{\Longrightarrow} & G_2 \end{array}$$

... then H_1 contains a possibly disconnected left-hand side which can be contracted via the edge contraction rules, resulting in H' and ...

GTS as Well-Structured Transition Systems

Obtaining a WSTS by adding edge contraction rules

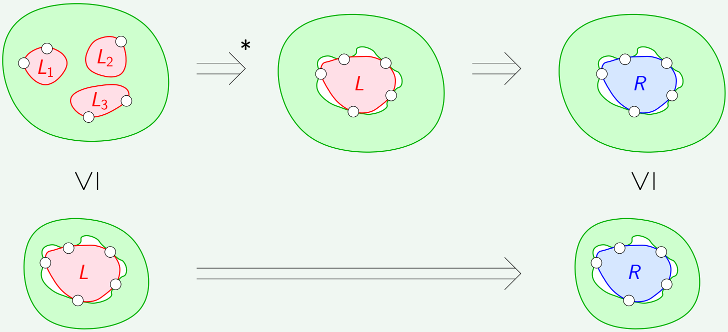
$$\begin{array}{ccc} H_1 \overset{*}{\Longrightarrow} H' \overset{r}{\Longrightarrow} H_2 \\ \vee\!| & & \vee\!| \\ G_1 \overset{r}{\Longrightarrow} G_2 \end{array}$$

... H' can be rewritten to H_2 (of which G_2 is a minor) by using the same rule as for G_1.

GTS as Well-Structured Transition Systems

$$
\begin{array}{ccccc}
H_1 & \Longrightarrow^* & H' & \stackrel{r}{\Longrightarrow} & H_2 \\
\begin{array}{c}\lor\!|\end{array} & & & & \begin{array}{c}\lor\!|\end{array} \\
G_1 & & \stackrel{r}{\Longrightarrow} & & G_2
\end{array}
$$

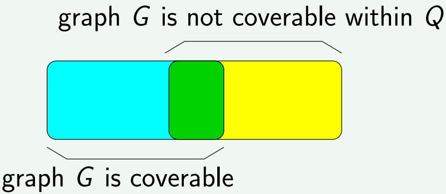
...



GTS as Well-Structured Transition Systems

What about the other orders (subgraph, induced subgraph)?

- The compatibility condition is satisfied (for arbitrary rules)
- We do not have a well-quasi-order on the set of all graphs
  $\leadsto$ Q-restricted WSTS
  - We still obtain decidability if Q is closed under reachability.
  - If the backwards analysis terminates on all graphs (no guarantee!) we still obtain correct results.
  - Otherwise we restrict the search space to Q and our method will give us one of the following two answers:



GTS as Well-Structured Transition Systems

order	wqo on Q	Q-res. well-structured
minor ordering	all graphs	lossy systems
subgraph ordering	bounded path length	GTS without NACs
ind. subgraph ordering	bounded path length and edge multiplicity	GTS with restricted NACs

Tradeoff:

- coarser order is potentially a well-quasi-order on a larger set of graphs.
- For a finer order more graph transformation systems can be well-structured.

Backward Analysis

What remains to be done in order to perform the backward analysis?

Given a finite basis F for an upward-closed set of graphs $\mathcal{U}$ we have to compute a finite basis for (the upward-closure of) $Pred(\mathcal{U})$.

Ideas:

- Given a graph $H \in F$, apply all rules backward.
- But: H need not contain the full right-hand side of a rule, but it may represent other graphs that do contain the right-hand side

  $\rightsquigarrow$ Instead of taking ordinary rules $r\colon L \rightharpoonup R$, take as rules $L \overset{r}{\rightharpoonup} R \overset{\mu}{\mapsto} M$, where μ is an arbitrary order morphism.

Backward Analysis

Why does this work?

Let $H \in \mathcal{U}$.

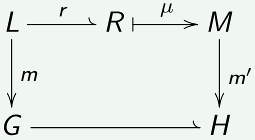
$$\begin{array}{ccccc} L & \xrightarrow{r} & R & \overset{\mu}{\longmapsto} & M \\ & & & & \big\downarrow{\scriptstyle m'} \\ & & & & H \end{array}$$

Find a match of M of the right-hand side in H.

Backward Analysis

Why does this work?

Let $H \in \mathcal{U}$.



Make a backward step by applying the rule backward (find a pushout complement).

Backward Analysis

Why does this work?

Let $H \in \mathcal{U}$.

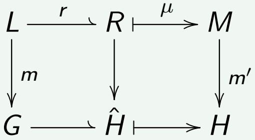
$$\begin{array}{ccccc} L & \xrightarrow{r} & R & \xrightarrow{\mu} & M \\ \downarrow m & & \downarrow & & \downarrow m' \\ G & \longrightarrow & \hat{H} & \longrightarrow & H \end{array}$$

This pushout splits into two pushouts (standard pushout splitting).
$\leadsto$ G can be rewritten to $\hat{H}$ and $H \leq \hat{H}$ (since order morphisms are preserved by pushouts).

Backward Analysis

Why does this work?

Let $H \in \mathcal{U}$.



$\Rightarrow$ $\hat{H} \in \mathcal{U}$ and $G \in Pred(\mathcal{U})$, i.e., the procedure is correct.

Backward Analysis

Why does this work?

Let $H \in \mathcal{U}$.

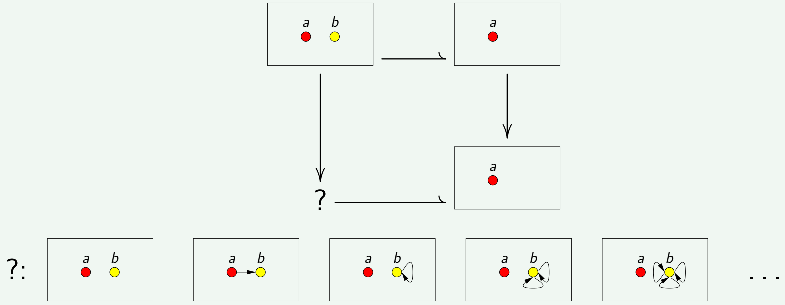
$$\begin{array}{ccccc} L & \xrightarrow{r} & R & \overset{\mu}{\longmapsto} & M \\ \downarrow{\scriptstyle m} & & \downarrow & & \downarrow{\scriptstyle m'} \\ G & \longrightarrow & \hat{H} & \longmapsto & H \end{array}$$

$\Rightarrow$ $\hat{H} \in \mathcal{U}$ and $G \in \mathit{Pred}(\mathcal{U})$, i.e., the procedure is correct.

Completeness, i.e., the fact that we generate the entire basis, also holds, but is more difficult to prove.

Backward Analysis

Another problem: in the category of partial morphisms, there are usually infinitely many pushout complements.



$\rightsquigarrow$ It is sufficient to compute only the minimal pushout complements with respect to the ordering. We have algorithms for this.

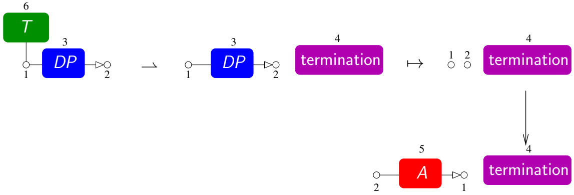
Backward Analysis

Backward analysis for the running example (minor ordering):

termination

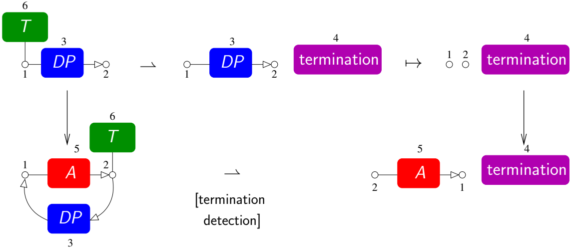
Backward Analysis

Backward analysis for the running example (minor ordering):



Backward Analysis

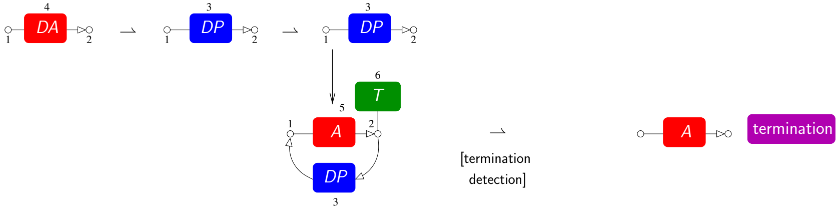
Backward analysis for the running example (minor ordering):



Apply rule [termination detection] backward.

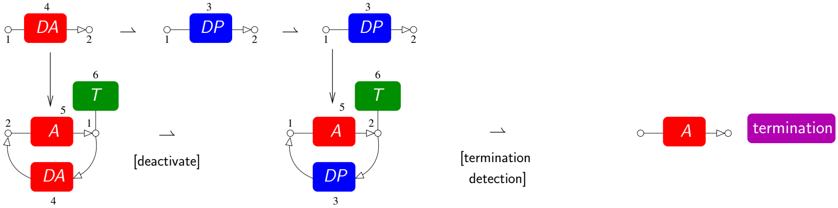
Backward Analysis

Backward analysis for the running example (minor ordering):



Backward Analysis

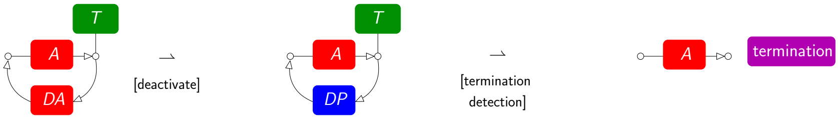
Backward analysis for the running example (minor ordering):



Apply rule [deactivate] backward.

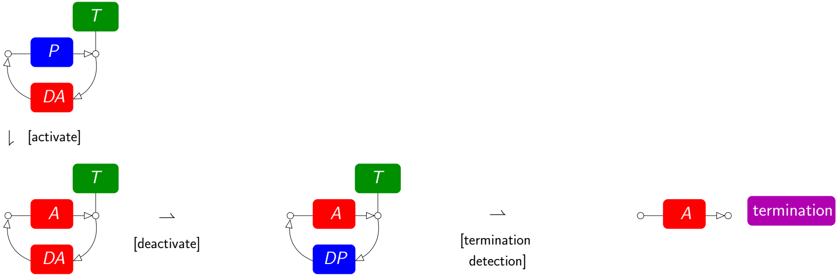
Backward Analysis

Backward analysis for the running example (minor ordering):



Backward Analysis

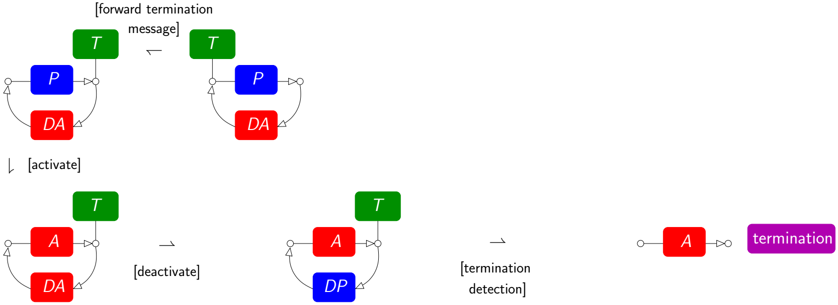
Backward analysis for the running example (minor ordering):



Apply rule [activate] backward.

Backward Analysis

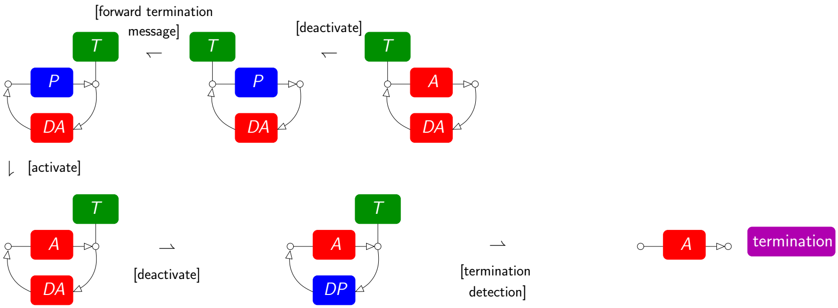
Backward analysis for the running example (minor ordering):



Apply rule [forward termination message] backward.

Backward Analysis

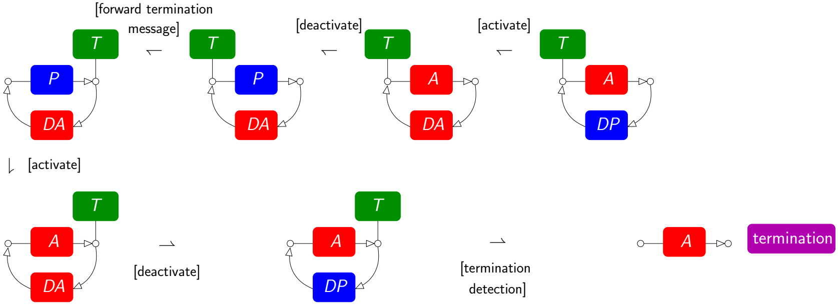
Backward analysis for the running example (minor ordering):



Apply rule [deactivate] backward.

Backward Analysis

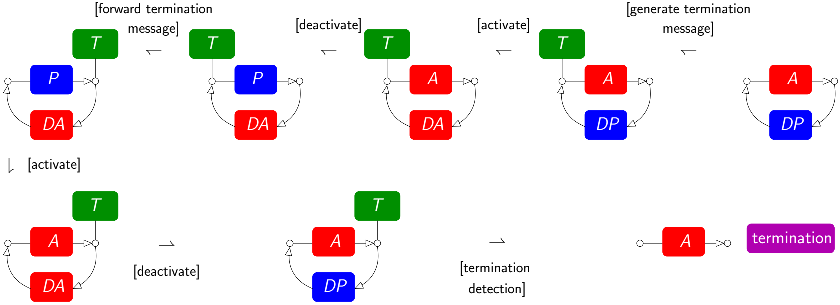
Backward analysis for the running example (minor ordering):



Apply rule [activate] backward.

Backward Analysis

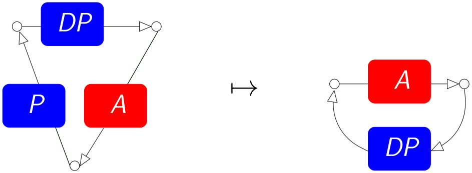
Backward analysis for the running example (minor ordering):



Apply rule [generate termination message] backward.

Backward Analysis

The last graph in this chain is a minor of the start graph!



This means that the error graph is indeed coverable and the termination detection rules are wrong.

Reason: after a passive detector sends a termination message he has to record whether he became again active (and then passive) before receiving this message

$\rightsquigarrow$ Rules have to be changed accordingly. Then the property can be verified (since this a decision procedure).

Implementation

Efficiency and Implementation

- We have a prototype implementation, based on the minor ordering and on the subgraph ordering.
- Runtime results:

case study	wqo	Q	time	#EG
Leader election	minor	all	< 1s	38
Termination det. (faulty)	minor	all	3s	69
Termination det. (correct)	minor	all	< 1s	101
Rights management	subg.	all	< 1s	4
Public-private server	subg.	path ≤ 6	1s	16
Public-private server	subg.	path ≤ 7	14s	18
Dining Philosophers	subg.	all	< 1s	12

Conclusion

Ongoing work

- Optimize and extend implementation, e.g. with the induced subgraph ordering.
- Universally quantified rules (allows to specify broadcasts and synchronization with neighbourhoods of arbitrary size) [Jan Stückrath, Giorgio Delzanno]

Future work

- Coarser orders preserving directed paths (topological minors?).
- Graph patterns instead of graphs [Saksena, Wibling, Jonsson]
- Forward analysis, cf. [Bansal, Koskinen, Wies, Zufferey].