Output-Determinacy and Asynchronous Circuit Synthesis

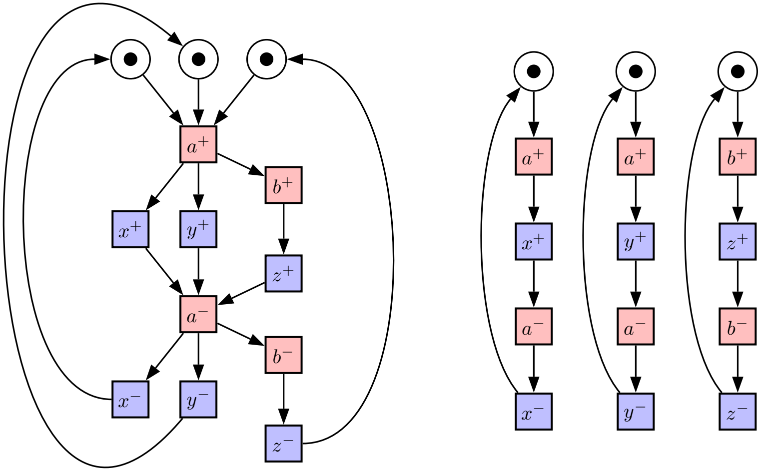
Victor Khomenko, Mark Schaefer, Walter Vogler

Newcastle University
School of Computing Science

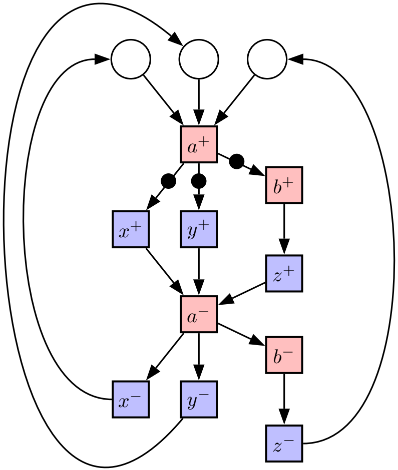
University of Augsburg
Institute of Computer Science

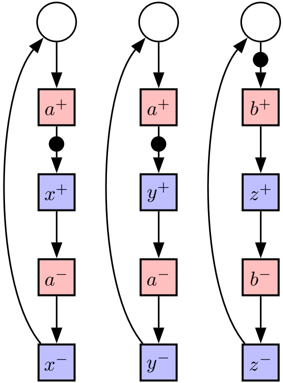
ACSD 2007

Decomposition and Correctness

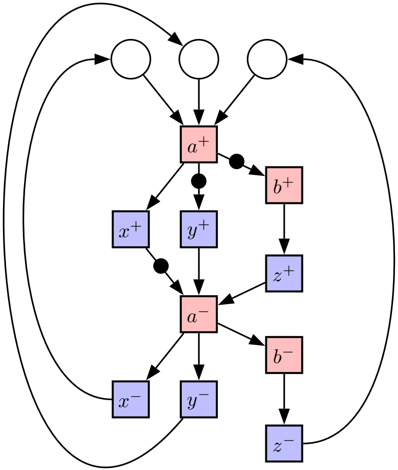


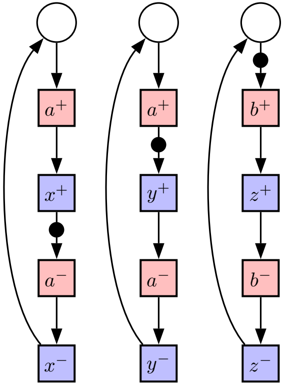
Decomposition and Correctness



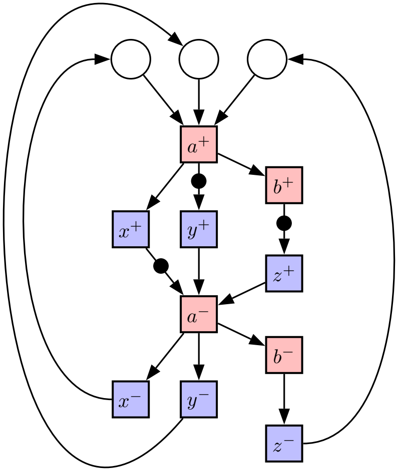


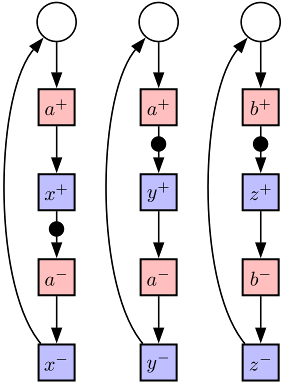
Decomposition and Correctness



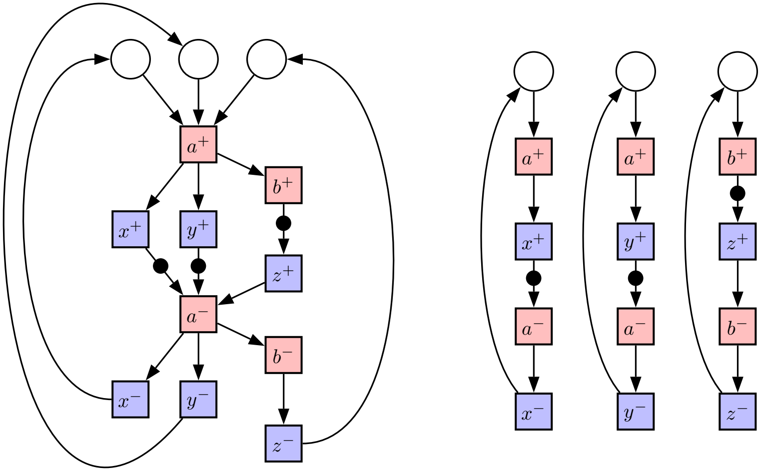


Decomposition and Correctness

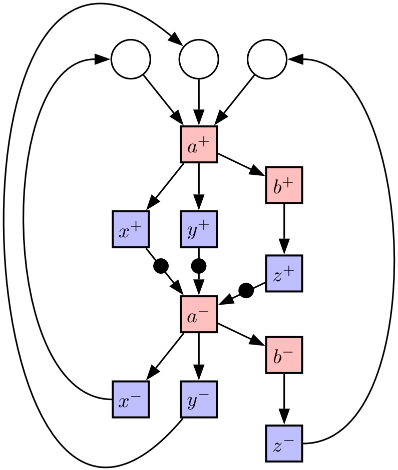


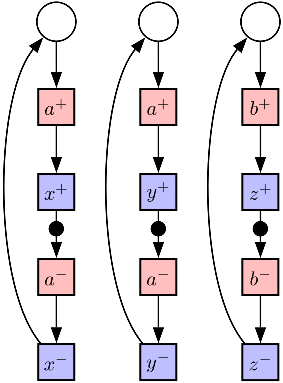


Decomposition and Correctness

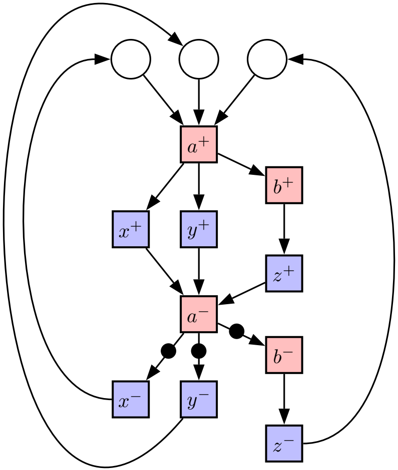


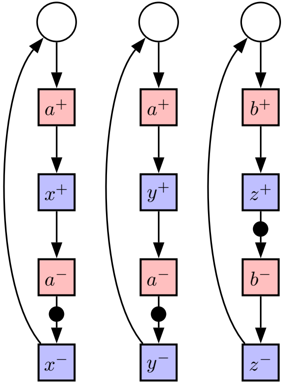
Decomposition and Correctness





Decomposition and Correctness



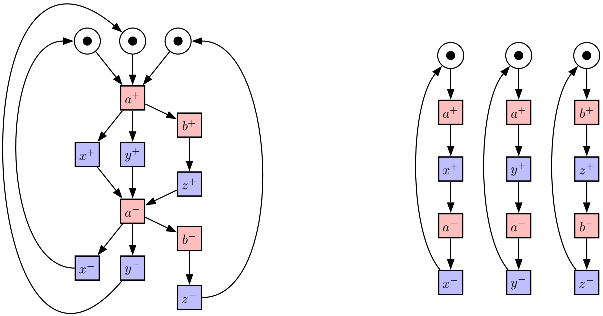


Advantage of Decomposition

- Concurrency is typical for asynchronous circuits, but leads to state explosion
- Synthesis needs a representation of this state space
- Decomposition generates smaller components with smaller state spaces

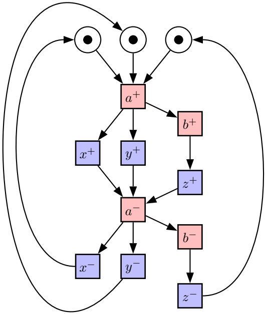
Advantage of Decomposition

- Concurrency is typical for asynchronous circuits, but leads to state explosion
- Synthesis needs a representation of this state space
- Decomposition generates smaller components with smaller state spaces

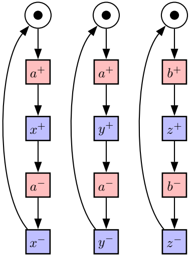


Advantage of Decomposition

- Concurrency is typical for asynchronous circuits, but leads to state explosion
- Synthesis needs a representation of this state space
- Decomposition generates smaller components with smaller state spaces

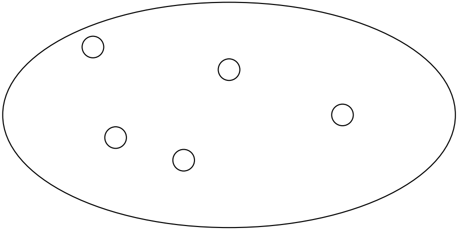


24 states

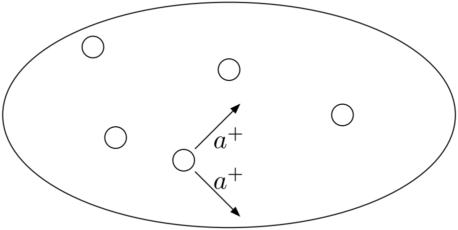


4 states each

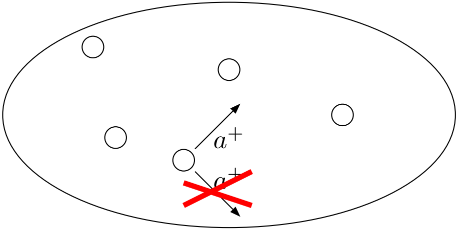
Determinism



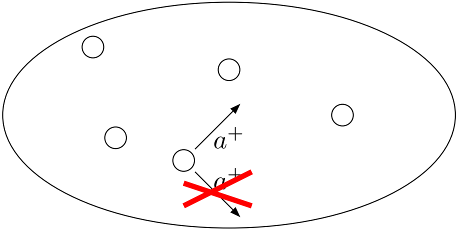
Determinism



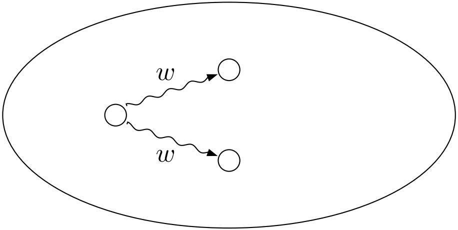
Determinism



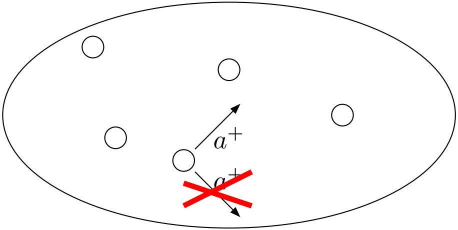
Determinism



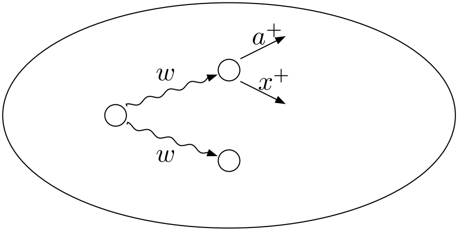
Determinacy



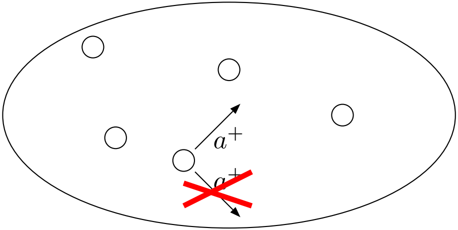
Determinism



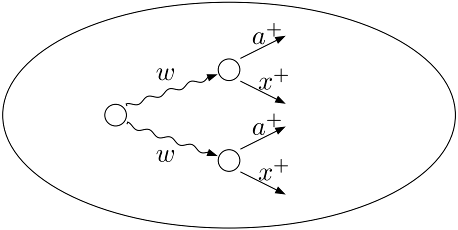
Determinacy



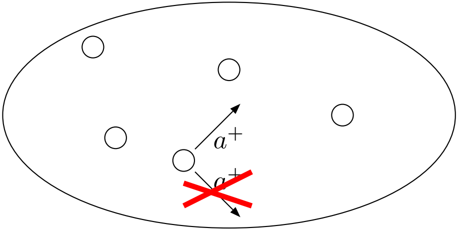
Determinism



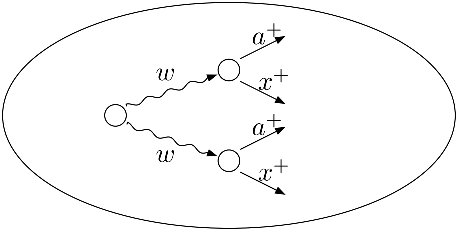
Determinacy

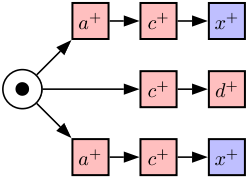


Determinism

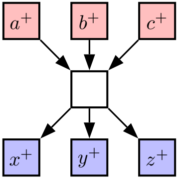


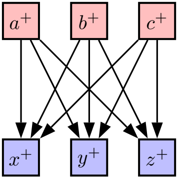
Determinacy





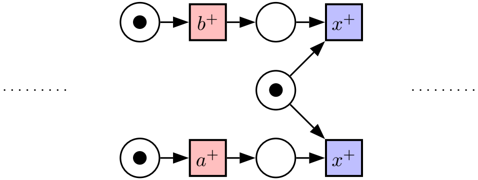
Dummy transitions are like λ/ϵ edges in an automaton.
Inserted for convenience.



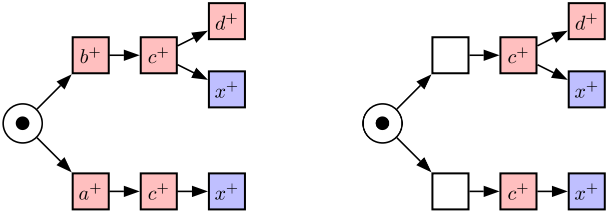


3 outputs are triggered by three inputs
left: 6 places and 1 transitions – right: 9 places

If an action x^+ should occur after the action a^+ **or** b^+ occurs, one has to use either unsafe or non-deterministic STGs. However, unsafe STGs are not preferable in praxis.

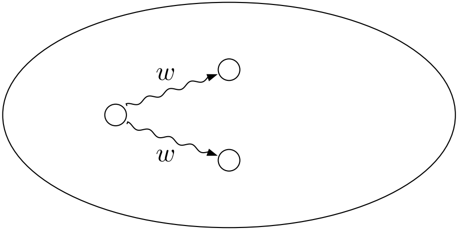


Hiding of signals, i.e. labelling them with λ. Essential operation of our STG decomposition algorithm.

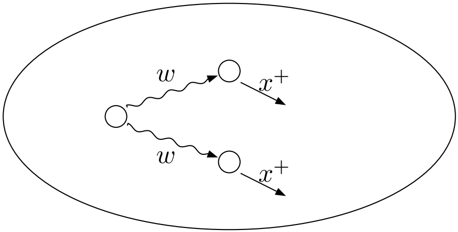


Hiding of *a* and *b*

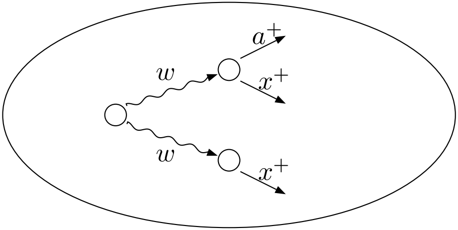
Output-Determinacy – Definition



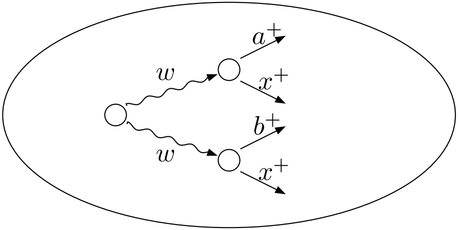
Output-Determinacy – Definition



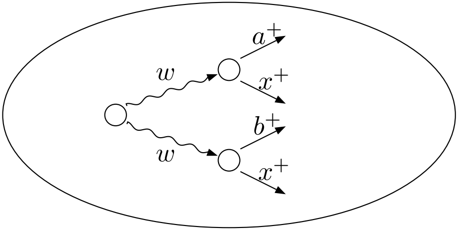
Output-Determinacy – Definition

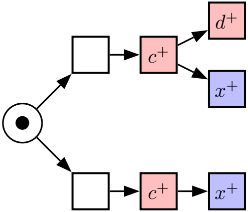


Output-Determinacy – Definition



Output-Determinacy – Definition





Output-Determinacy – Why not Determinise?

- Non-deterministic specifications can be **determinised**, i.e. transformed into a deterministic one with the same language.

Output-Determinacy – Why not Determinise?

- Non-deterministic specifications can be **determinised**, i.e. transformed into a deterministic one with the same language.
- For deterministic specifications the language is an adequate semantics

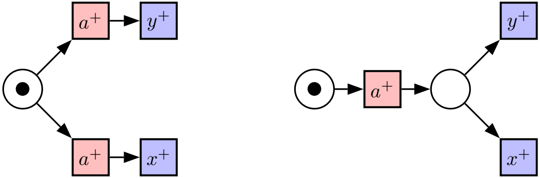
Output-Determinacy – Why not Determinise?

- Non-deterministic specifications can be **determinised**, i.e. transformed into a deterministic one with the same language.
- For deterministic specifications the language is an adequate semantics
- However, this leads to problems if the original specification is not output-determinate:

Output-Determinacy – Why not Determinise?

- Non-deterministic specifications can be **determinised**, i.e. transformed into a deterministic one with the same language.
- For deterministic specifications the language is an adequate semantics
- However, this leads to problems if the original specification is not output-determinate:
  - Violation of semi-modularity
  - Violation of correctness

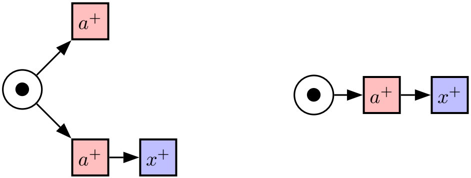
Violation of Semi-Modularity



left: non-output-determinate: STG
right: determinised version, x^+ is in conflict with y^+

Output-Determinacy – Why not Determinise?

Violation of Correctness



left: non-output-determinate: STG
right: determinised version, can be synthesised but is not correct

Output-Determinacy – CSC Conflicts

- Non-output-determinacy is also a CSC conflict
- CSC conflicts can be resolved by insertion of internal signals in a behaviour-preserving way
- Does not work for violation of output-determinacy

Output-Determinacy – Justification

A non-output-determinate specification is ill-formed and cannot be implemented speed-independently.

Output-Determinacy – Justification

A non-output-determinate specification is ill-formed and cannot be implemented speed-independently.

The semantics of an output-determinate speed-independent specification is its language.

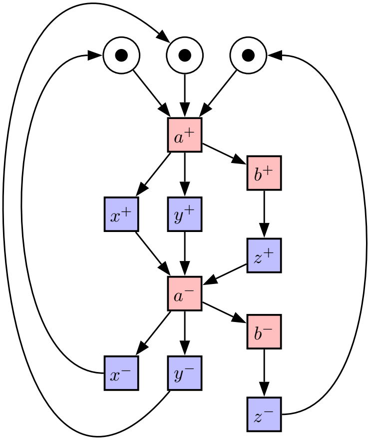
Output-Determinacy – Justification

A non-output-determinate specification is ill-formed and cannot be implemented speed-independently.

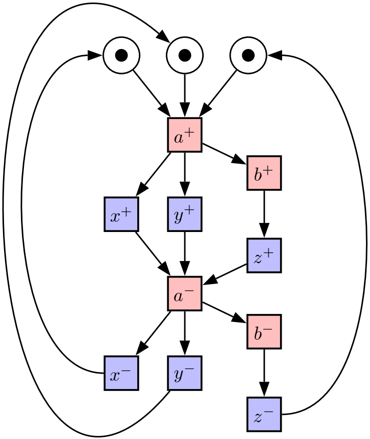
The semantics of an output-determinate speed-independent specification is its language.

These give a semantics for non-deterministic specifications.

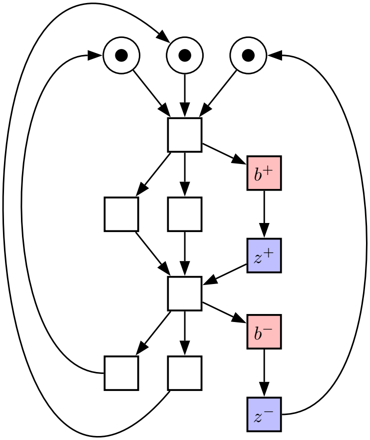
Decomposition – Example

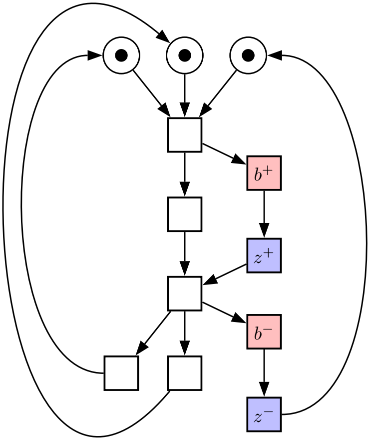


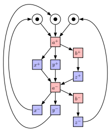
Decomposition – Example



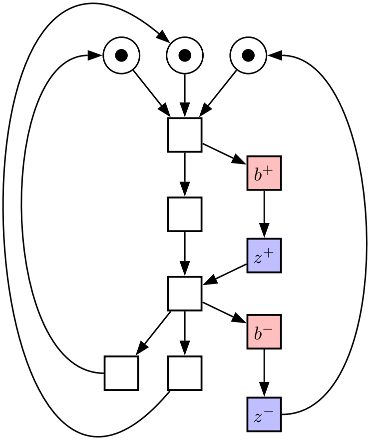
Decomposition – Example

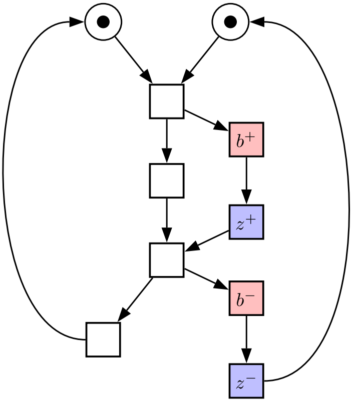


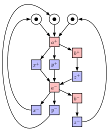


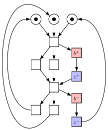


Decomposition – Example

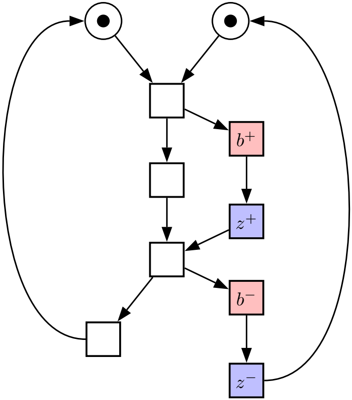


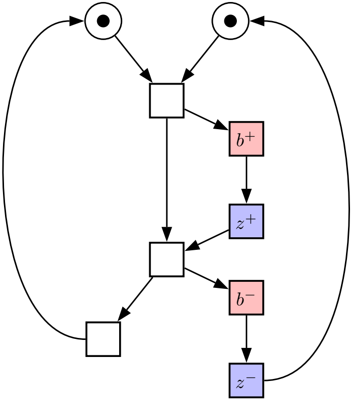


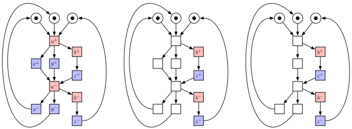




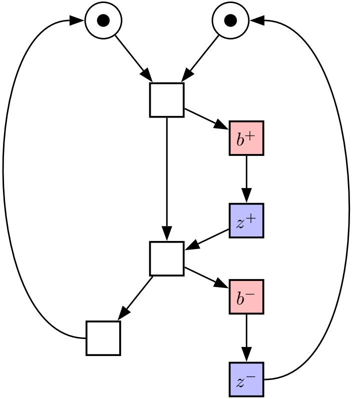
Decomposition – Example

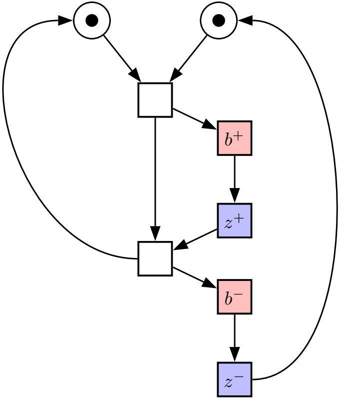


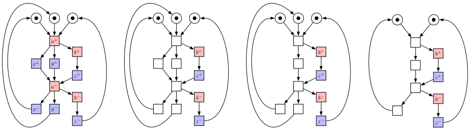




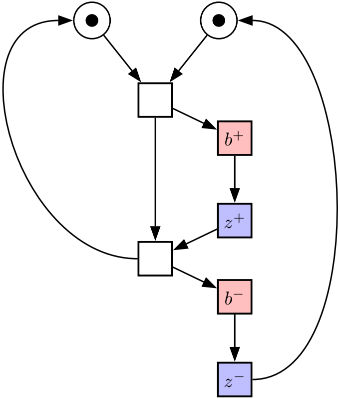
Decomposition – Example

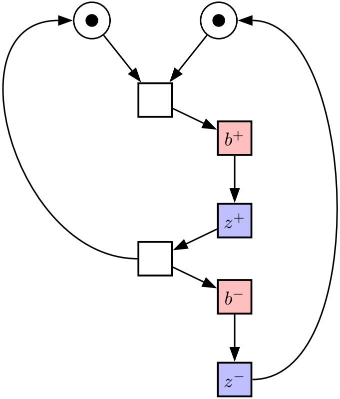


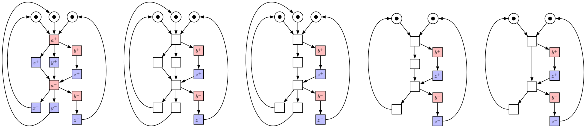




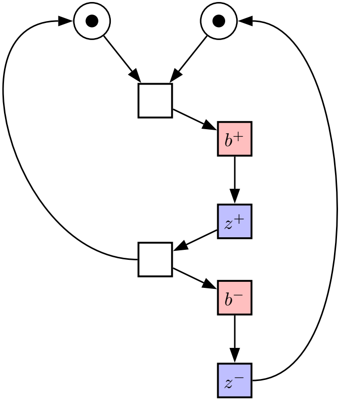
Decomposition – Example

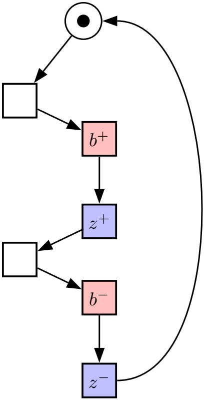


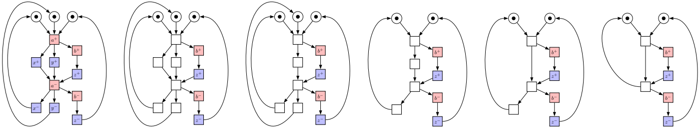




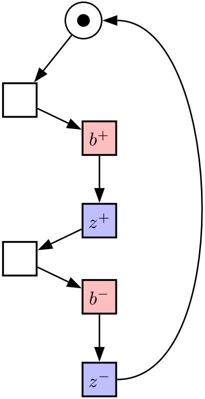
Decomposition – Example

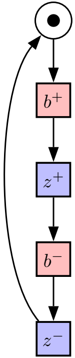


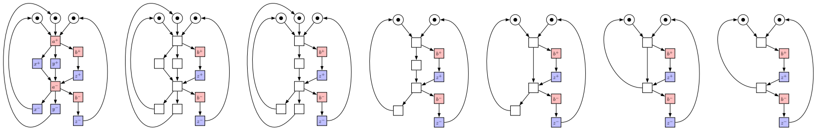




Decomposition – Example







Old Approach

- Decomposition is not always so easy: often, it is impossible to remove all λ transitions introduced before

Old Approach

- Decomposition is not always so easy: often, it is impossible to remove all λ transitions introduced before
- Then: **backtracking**, i.e. restarting at the initial component with additional signals

Decomposition – Backtracking

Old Approach

- Decomposition is not always so easy: often, it is impossible to remove all λ transitions introduced before
- Then: **backtracking**, i.e. restarting at the initial component with additional signals
- Either necessary to get a correct component, or just for technical reasons

Decomposition – Backtracking

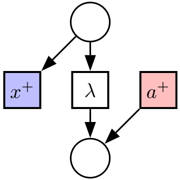
Old Approach

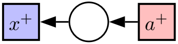
- Decomposition is not always so easy: often, it is impossible to remove all λ transitions introduced before
- Then: **backtracking**, i.e. restarting at the initial component with additional signals
- Either necessary to get a correct component, or just for technical reasons

New Approach

- No backtracking for technical reasons
- λ transitions can be left in the component, as long as it is output-determinate (checked implicitly during synthesis)

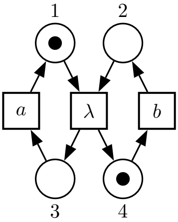
Decomposition – Non-Secure Contraction

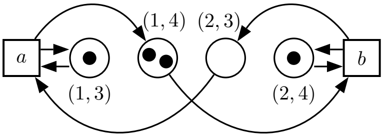




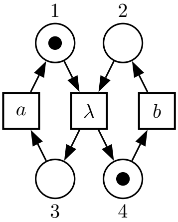
Language is changed

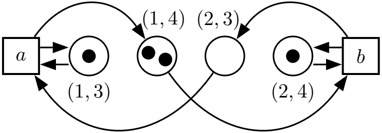
Decomposition – Non Safeness-Preserving Contraction





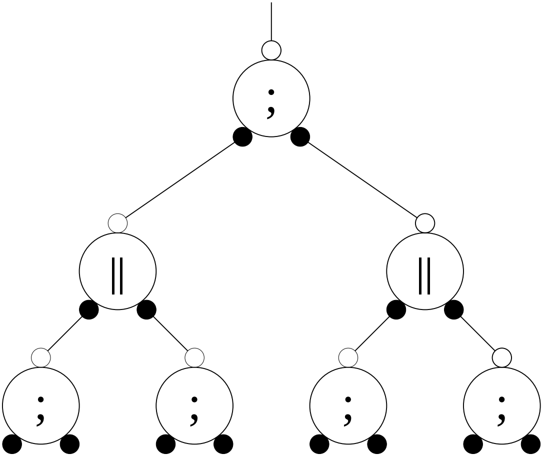
Decomposition – Non Safeness-Preserving Contraction





- Synthesis tools do not support non-safe STGs
- or just with a place capacity
- Latter is inefficient: every contraction doubles potential place capacity

Benchmarks — SeqParTree



Artificial Balsa Handshake component tree
of sequencers and parallelisers with 3 Levels

Benchmarks — Results

Level	$\lvert P\rvert - \lvert T\rvert$	$\lvert In\rvert - \lvert Out\rvert$	Old	New	Synthesis
5	382 – 252	33 – 93	1	1	5
6	798 – 508	65 – 189	4	4	16
7	1566 – 1020	129 – 381	9	8	22
8	3230 – 2044	257 – 765	32	17	1:02
9	6302 – 4092	513 – 1533	1:27	1:18	1:30
10	12958 – 8188	1025 – 3069	42:37	6:03	4:32

Level	$\|P\| - \|T\|$	$\|In\| - \|Out\|$	Old	New	Synthesis
5	382 – 252	33 – 93	1	1	5
6	798 – 508	65 – 189	4	4	16
7	1566 – 1020	129 – 381	9	8	22
8	3230 – 2044	257 – 765	32	17	1:02
9	6302 – 4092	513 – 1533	1:27	1:18	1:30
10	12958 – 8188	1025 – 3069	42:37	6:03	4:32

Synthesis of an STG with 4094 signals in about 11 minutes.

More Results in the Paper

Deciding Output-Determinacy . . .

- PSPACE complete for bounded/safe STGs
- EXPSPACE hard for unbounded STGs

More Results in the Paper

Deciding Output-Determinacy . . .

- PSPACE complete for bounded/safe STGs
- EXPSPACE hard for unbounded STGs

Decomposition:

- Simplified correctness notion
- Decomposition can be applied to output-determinate (possibly) non-deterministic specifications
- More decomposition operations

Conclusion

Introduced output-determinacy

- Relaxation of determinism and determinacy
- Fits very well with STGs and speed-independency
- Allows a language-based semantics

Conclusion

Introduced output-determinacy

- Relaxation of determinism and determinacy
- Fits very well with STGs and speed-independency
- Allows a language-based semantics

Applied output-determinacy to STG decomposition

- Decomposition gets faster
- Allows more STG specifications
- More decomposition operations

Future Research and Related Work

Future Reseach

- Semantics for internal signals – application to output-determinacy and decomposition
- Decomposition works structurally only.
  Fast but problems with 'tricky' specifications
- Apply decomposition to BALSA resynthesis
- Preserving CSC in the components

Future Research and Related Work

Future Reseach

- Semantics for internal signals – application to output-determinacy and decomposition
- Decomposition works structurally only.
  Fast but problems with ‘tricky’ specifications
- Apply decomposition to BALSA resynthesis
- Preserving CSC in the components

Related Work

- Carmona and Cortadella: ILP approach, e.g
  ILP models for the synthesis of asynchronous control circuits
- Myers and Yoneda: structural decomposition, e.g.
  Synthesis of Speed Independent Circuits Based on Decomposition