Confluence up to Garbage

13th International Conference on Graph Transformation

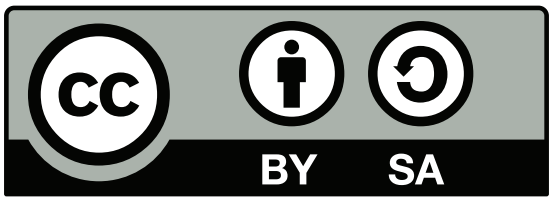
Graham Campbell

School of Mathematics, Statistics and Physics, Newcastle University, UK

Detlef Plump

Department of Computer Science, University of York, UK

June 2020



Motivating Question

When does a graph grammar admit a deterministic algorithm for deciding membership?

Graph Languages

Setting: DPO graph transformation with:

- all rules and matches are injective;
- systems with finitely many rules.

Definition (Graph Grammar)

Given a GT system $T = (\Sigma, \mathcal{R})$, a subalphabet of *non-terminals* $\mathcal{N}$, and a *start graph* S over Σ, then a grammar is a tuple $(\Sigma, \mathcal{N}, \mathcal{R}, S)$.

$$\mathrm{L}(\mathcal{G}) = \{G \mid S \Rightarrow^*_{\mathcal{R}} G, G \text{ terminally labelled}\}.$$

Graph Languages

Setting: DPO graph transformation with:

- all rules and matches are injective;
- systems with finitely many rules.

Definition (Graph Grammar)

Given a GT system $T = (\Sigma, \mathcal{R})$, a subalphabet of *non-terminals* $\mathcal{N}$, and a *start graph* S over Σ, then a grammar is a tuple $(\Sigma, \mathcal{N}, \mathcal{R}, S)$.

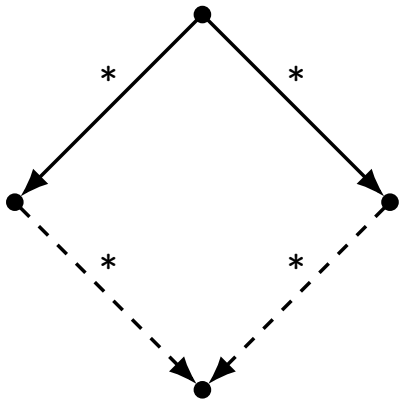
$$\mathrm{L}(\mathcal{G}) = \{G \mid S \Rightarrow^*_{\mathcal{R}} G, G \text{ terminally labelled}\}.$$

- $G \Rightarrow_r H$ if and only if $H \Rightarrow_{r^{-1}} G$, for some $r \in \mathcal{R}$.
- $G \in \mathrm{L}(\mathcal{G})$ if and only if $G \Rightarrow^*_{\mathcal{R}^{-1}} S$ and G is terminally labelled.
- If $\mathcal{R}^{-1}$ terminates, then we have a non-deterministic membership checking algorithm.

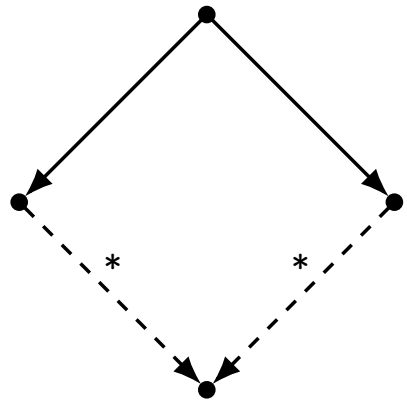
Example: Recognising Trees

Starting from a single node, the following DPO rule can be used to generate the language of trees:

The reversed system is confluent and terminating, so no backtracking is required to check if a graph is a tree.



(a) Confluence



(b) Local Confluence

Results about Confluence

Theorem (Plump (1998) and Plump (2005))

It is undecidable, in general:

- whether a GT system is terminating;
- whether a terminating GT system is confluent.

Lemma (Plump (2005))

A GT system is locally confluent if (but not only if) all its critical pairs are strongly joinable.

Results about Confluence

Theorem (Plump (1998) and Plump (2005))

It is undecidable, in general:

- whether a GT system is terminating;
- whether a terminating GT system is confluent.

Lemma (Plump (2005))

A GT system is locally confluent if (but not only if) all its critical pairs are strongly joinable.

Lemma (Newman (1942))

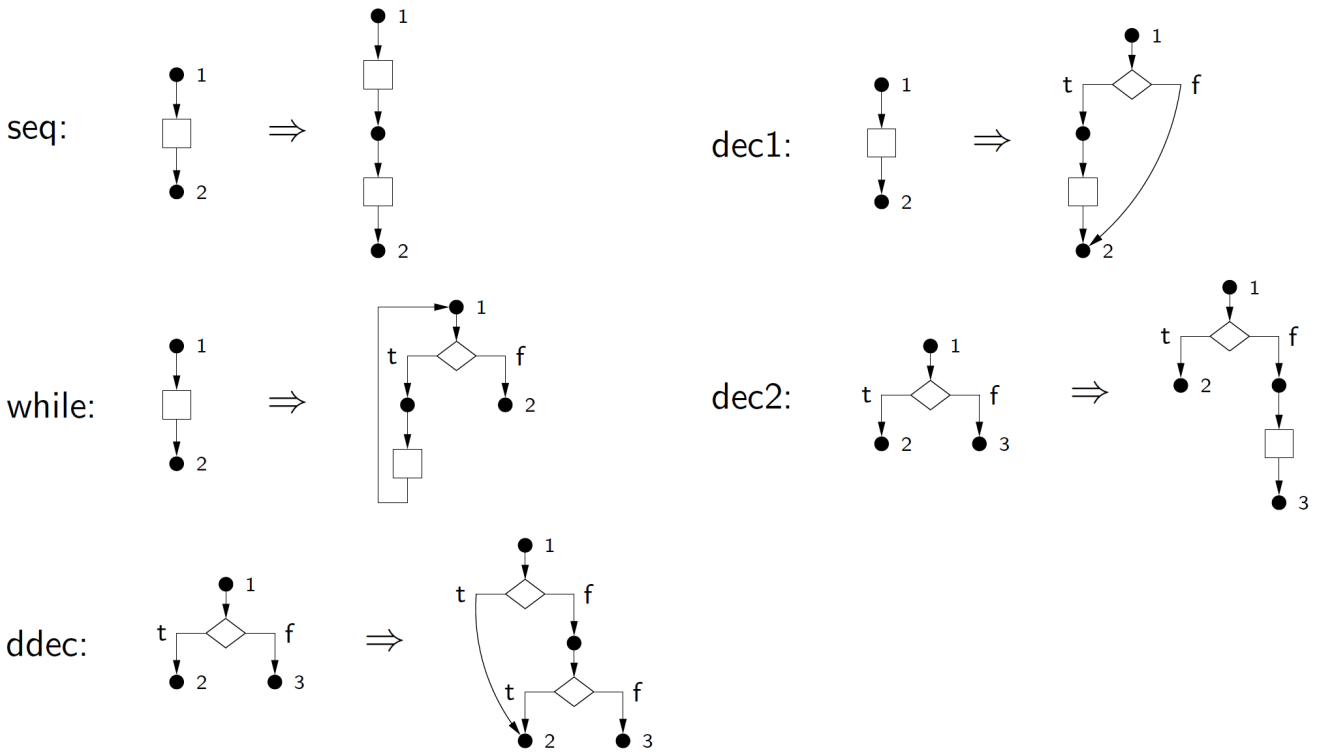
In the presence of termination, confluence and local confluence coincide.

Given termination, we may be able to automatically detect confluence.

Example: Extended Flow Diagrams I

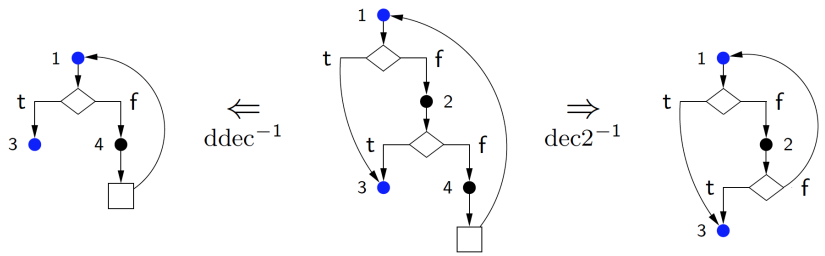
The language of EFDs is a is contained in the language semi-structured flow graphs (Farrow, Kennedy, and Zucconi 1976; Plump 2005).

The language is generated by:



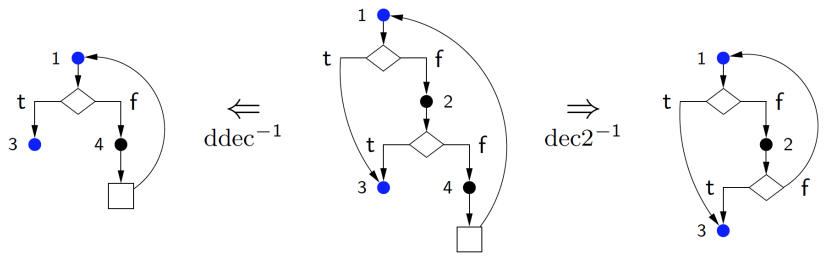
Example: Extended Flow Diagrams II

The inverse rules are not confluent. Non-joinable pair:

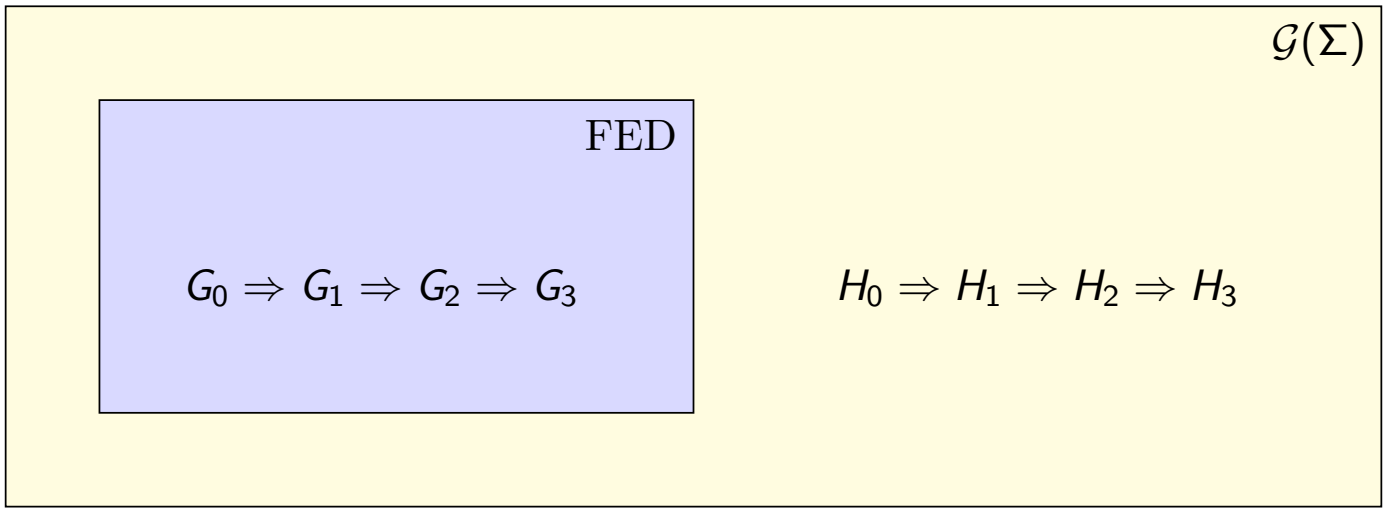


Example: Extended Flow Diagrams II

The inverse rules are not confluent. Non-joinable pair:

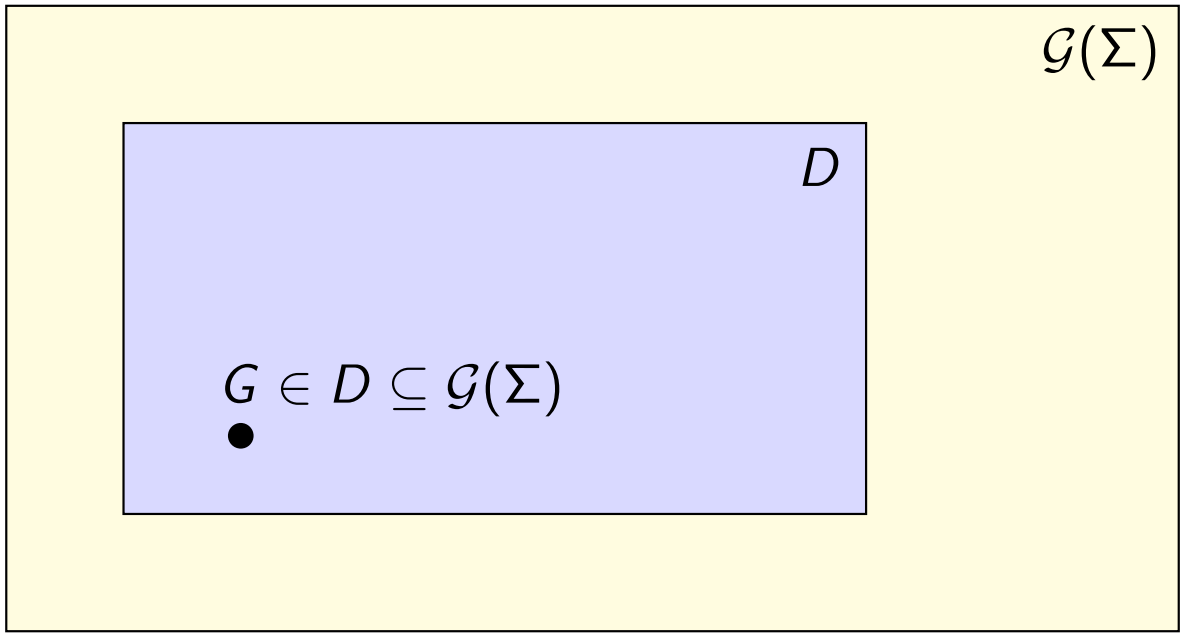


It will turn out that we still have deterministic membership checking:



Confluence up to Garbage

Confluence up to garbage only considers start graphs in some language D.



Theorem

Given a grammar $\mathcal{G}$ with rules $\mathcal{R}$, termination and confluence up to garbage of $\mathcal{R}^{-1}$ give us deterministic membership checking for $\mathrm{L}(\mathcal{G})$.

Non-Garbage Critical Pairs

Definition (Critical Pair)

A critical pair of $(\Sigma, \mathcal{R})$ is a pair of parallelly dependent direct derivations $H_1 \Leftarrow_{r_1,g_1} G \Rightarrow_{r_2,g_2} H_2$ such that $G = g_1(L_1) \cup g_2(L_2)$ and if $r_1 = r_2$ then $g_1 \neq g_2$.

Non-Garbage Critical Pairs

Definition (Critical Pair)

A critical pair of $(\Sigma, \mathcal{R})$ is a pair of parallelly dependent direct derivations $H_1 \Leftarrow_{r_1,g_1} G \Rightarrow_{r_2,g_2} H_2$ such that $G = g_1(L_1) \cup g_2(L_2)$ and if $r_1 = r_2$ then $g_1 \neq g_2$.

Definition (Non-Garbage Critical Pair)

Let $T = (\Sigma, \mathcal{R})$ and $\mathcal{D} \subseteq \mathcal{G}(\Sigma)$. A $\mathcal{D}$-non-garbage critical pair of T is a critical pair $H_1 \Leftarrow_{r_1,g_1} G \Rightarrow_{r_2,g_2} H_2$ such that G is in $\widehat{\mathcal{D}}$, the subgraph closure of $\mathcal{D}$.

If $\mathcal{D}$ has decidable subgraph closure, then we can effectively enumerate the (finitely many) non-garbage critical pairs.

Generalised Critical Pair Lemma I

Definition (Strong Joinability)

A critical pair is strongly joinable if it is joinable without deleting any of the persistent nodes, and the persistent nodes are identified when joining.

Generalised Critical Pair Lemma I

Definition (Strong Joinability)

A critical pair is strongly joinable if it is joinable without deleting any of the persistent nodes, and the persistent nodes are identified when joining.

Theorem (Generalised Critical Pair Lemma)

Let $\mathcal{T} = (\Sigma, \mathcal{R})$ and $\mathcal{D} \subseteq \mathcal{G}(\Sigma)$. If all $\mathcal{T}$'s $\mathcal{D}$-non-garbage critical pairs are strongly joinable, then $\mathcal{T}$ is locally confluent on $\mathcal{D}$.

Generalised Critical Pair Lemma I

Definition (Strong Joinability)

A critical pair is strongly joinable if it is joinable without deleting any of the persistent nodes, and the persistent nodes are identified when joining.

Theorem (Generalised Critical Pair Lemma)

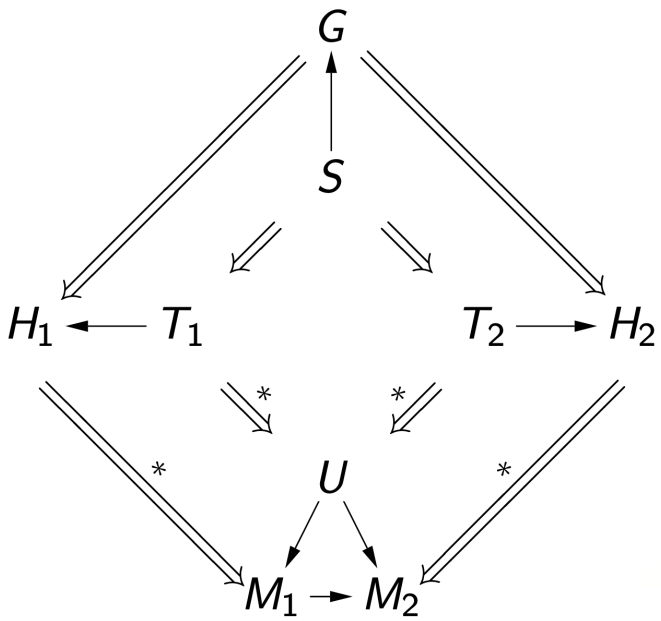
Let $T = (\Sigma, \mathcal{R})$ and $\mathcal{D} \subseteq \mathcal{G}(\Sigma)$. If all T's $\mathcal{D}$-non-garbage critical pairs are strongly joinable, then T is locally confluent on $\mathcal{D}$.

Proof sketch: We need to show that every pair of derivations $H_1 \Leftarrow_{r_1,g_1} G \Rightarrow_{r_2,g_2} H_2$ such that G is non-garbage can be joined.

Case 1: If parallelly independent, then done by commutativity (Ehrig and Kreowski 1976).

Generalised Critical Pair Lemma II

Case 2: The Clipping Theorem (Kreowski 1977) tells us how to factor out a minimal derivation $T_1 \Leftarrow S \Rightarrow T_2$. This must be a critical pair, and $G \in \mathcal{D}$, then $S \in \widehat{\mathcal{D}}$, so the critical pair is non-garbage. Non-garbage pairs are strongly joinable (by assumption), so we can use the Embedding Theorem (Ehrig 1977) to get our result.

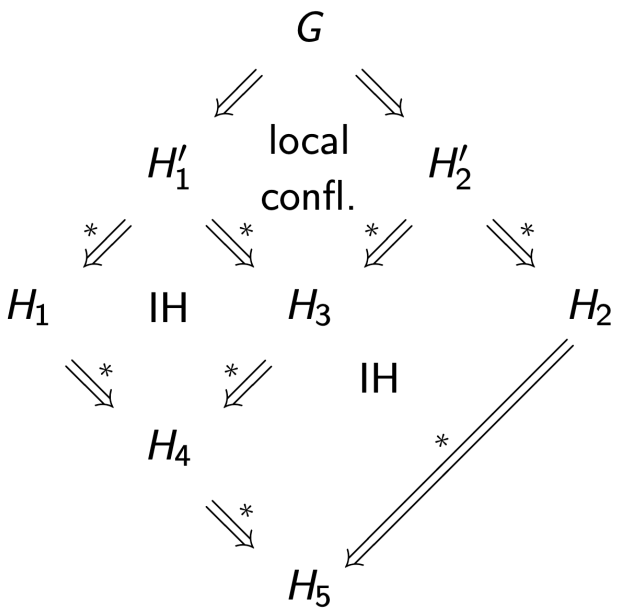


Generalised Newman's Lemma

Lemma (Generalised Newman's Lemma)

Let $T = (\Sigma, \mathcal{R})$ and $\mathcal{D} \subseteq \mathcal{G}(\Sigma)$. If T is terminating on $\mathcal{D}$ and $\mathcal{D}$ is closed under T, then T is confluent on $\mathcal{D}$ if and only if is locally confluent on $\mathcal{D}$.

Proof sketch: Noetherian induction.



Checking for Confluence up to Garbage

Input:

1. GT system T.
2. Language $\mathcal{D}$ (possibly specified by a grammar).

Checking for Confluence up to Garbage

Input:

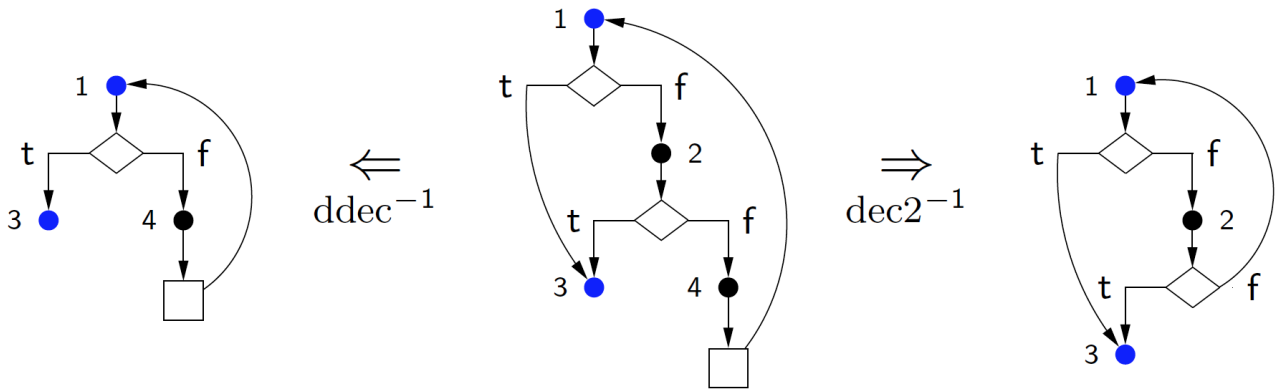
1. GT system T.
2. Language $\mathcal{D}$ (possibly specified by a grammar).

Process:

1. Establish that T is terminating on $\mathcal{D}$ and $\mathcal{D}$ is closed under T. If closedness cannot be established, restart with a superset of D.
2. Generate the finitely many non-garbage critical pairs of T.
3. Check if each generated pair is strongly joinable:
   - All pairs are strongly joinable: confluence on $\mathcal{D}$;
   - One of the pairs is not joinable: a direct counter example;
   - All pairs are joinable, but not all strongly: inconclusive.

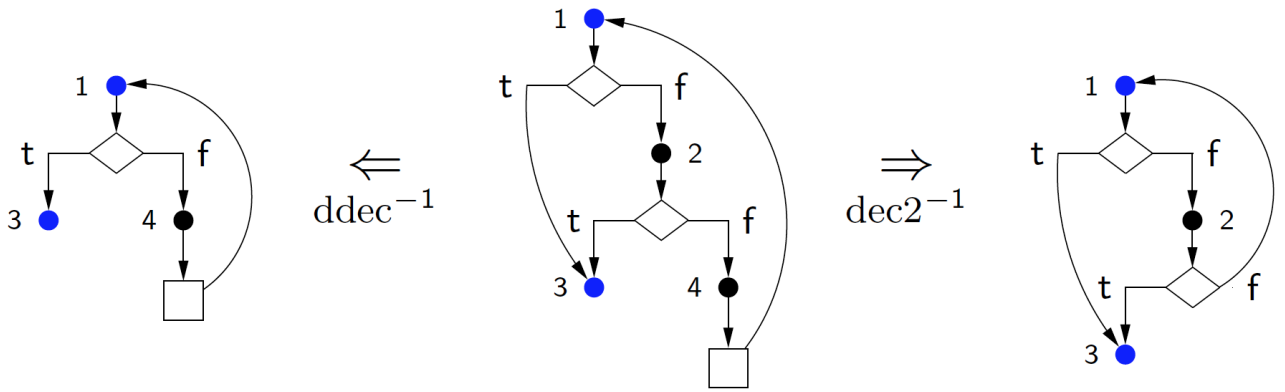
Example: Extended Flow Diagrams III

Recall we had a non-joinable pair:



Example: Extended Flow Diagrams III

Recall we had a non-joinable pair:



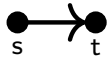
- The start graph of the pair cannot be embedded in an EFD.
- The pair is garbage, and all others are strongly joinable.
- We have termination and confluence up to garbage.
- Thus, we have determinsitic recognition.

Example: Series Parallel Graphs I

Series-parallel graphs were introduced by Duffin (1965). It is the smallest class of graphs closed under *parallel* and *sequential* composition, containing $P = \overset{s}{\bullet}\!\!\rightarrow\!\!\overset{t}{\bullet}$, where s is the *source* and t the *sink*.

Parallel composition identifies the two *sources* and *sinks*, and *sequential* composition identifies the *sink* of one with the *source* of another.

Example: Series Parallel Graphs I

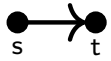
Series-parallel graphs were introduced by Duffin (1965). It is the smallest class of graphs closed under *parallel* and *sequential* composition, containing $P =$ 

, where s is the *source* and t the *sink*.

Parallel composition identifies the two *sources* and *sinks*, and *sequential* composition identifies the *sink* of one with the *source* of another.

Duffin showed that a graph is series-parallel if and only if it can be generated from P by the DPO rules:

Example: Series Parallel Graphs I

Series-parallel graphs were introduced by Duffin (1965). It is the smallest class of graphs closed under *parallel* and *sequential* composition, containing $P =$ 

, where s is the *source* and t the *sink*.

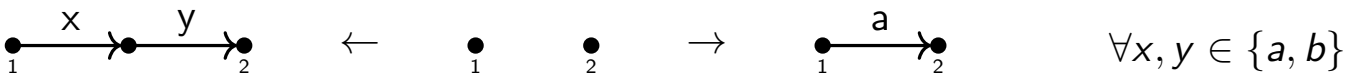
Parallel composition identifies the two *sources* and *sinks*, and *sequential* composition identifies the *sink* of one with the *source* of another.

Duffin showed that a graph is series-parallel if and only if it can be generated from P by the DPO rules:

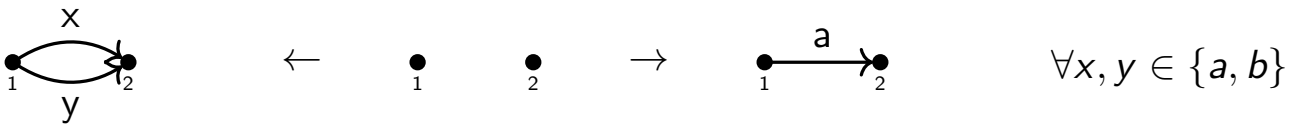
The reversed rules are confluent and terminating, however introducing labels to the edges breaks confluence (Hristakiev and Plump 2018), but we can still establish confluence up to garbage...

Example: Series Parallel Graphs II

Reduction rules for the language of labelled series-parallel graphs:

s_i: 

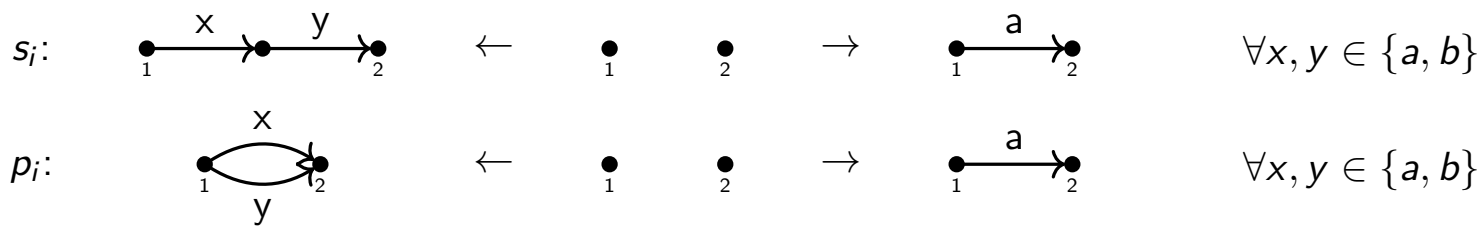
$\forall x, y \in \{a, b\}$

p_i: 

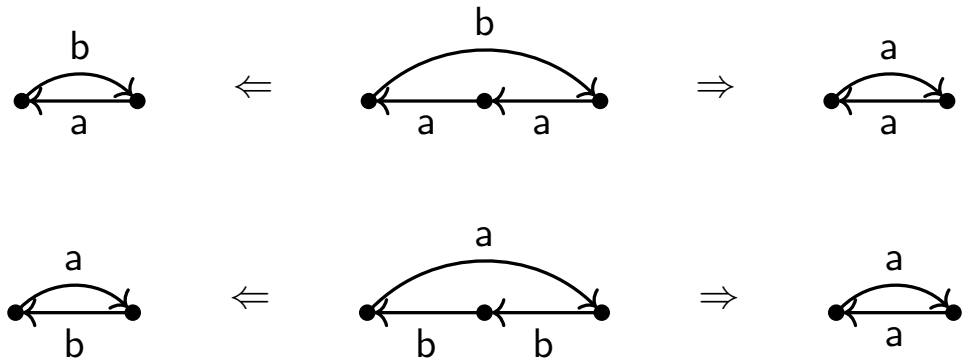
$\forall x, y \in \{a, b\}$

Example: Series Parallel Graphs II

Reduction rules for the language of labelled series-parallel graphs:



All critical pairs are strongly joinable other than the garbage ones:



We have termination and confluence up to garbage, so determinsitic recognition, as before.

Generalising The Results

- The examples in this talk only showed grammars without non-terminals, but the process applies in general.

Generalising The Results

- The examples in this talk only showed grammars without non-terminals, but the process applies in general.
- The Generalised Newman's Lemma works for any ARS.

Generalising The Results

- The examples in this talk only showed grammars without non-terminals, but the process applies in general.
- The Generalised Newman's Lemma works for any ARS.
- The Generalised Critical Pair Lemma works:
  - for (labelled) hypergraphs too, and we can relax injectivity of rules such that $K \to R$ need not be injective;
  - for any adhesive system with the usual restrictions, though the proof is a little different (Campbell 2019);
  - in the setting of (hyper)graph transformation with node relabelling (with root nodes) (not ($\mathcal{M}$-)adhesive!) (Campbell and Plump 2019).

Related and Future Work

- Usefulness of classification of non-garbage conflicts (Lambers, Born, Orejas, Strüber, and Taentzer 2018; Lambers, Kosiol, Strüber, and Taentzer 2019)?

Related and Future Work

- Usefulness of classification of non-garbage conflicts (Lambers, Born, Orejas, Strüber, and Taentzer 2018; Lambers, Kosiol, Strüber, and Taentzer 2019)?
- Connection to essential critical pairs (Lambers, Ehrig, and Orejas 2008; Lambers, Born, Orejas, Strüber, and Taentzer 2018)?

Related and Future Work

- Usefulness of classification of non-garbage conflicts (Lambers, Born, Orejas, Strüber, and Taentzer 2018; Lambers, Kosiol, Strüber, and Taentzer 2019)?
- Connection to essential critical pairs (Lambers, Ehrig, and Orejas 2008; Lambers, Born, Orejas, Strüber, and Taentzer 2018)?
- Connection to graphs satisfying negative constraints (Lambers 2009)?

Related and Future Work

- Usefulness of classification of non-garbage conflicts (Lambers, Born, Orejas, Strüber, and Taentzer 2018; Lambers, Kosiol, Strüber, and Taentzer 2019)?
- Connection to essential critical pairs (Lambers, Ehrig, and Orejas 2008; Lambers, Born, Orejas, Strüber, and Taentzer 2018)?
- Connection to graphs satisfying negative constraints (Lambers 2009)?
- Find checkable sufficient conditions which allow to decide membership in the subgraph closure of a language.

Related and Future Work

- Usefulness of classification of non-garbage conflicts (Lambers, Born, Orejas, Strüber, and Taentzer 2018; Lambers, Kosiol, Strüber, and Taentzer 2019)?
- Connection to essential critical pairs (Lambers, Ehrig, and Orejas 2008; Lambers, Born, Orejas, Strüber, and Taentzer 2018)?
- Connection to graphs satisfying negative constraints (Lambers 2009)?
- Find checkable sufficient conditions which allow to decide membership in the subgraph closure of a language.
- Connection to confluence modulo garbage (work in progress).

References I

Campbell, Graham (2019). "Efficient Graph Rewriting". BSc Thesis. Department of Computer Science, University of York, UK.

Campbell, Graham and Detlef Plump (2019). *Efficient Recognition of Graph Languages*. Tech. rep. Department of Computer Science, University of York, UK.

Duffin, R. J. (1965). "Topology of Series-Parallel Networks". In: *Journal of Mathematical Analysis and Applications* 10.2, pp. 303–318. DOI: 10.1016/0022-247X(65)90125-3.

Ehrig, Hartmut (1977). "Embedding theorems in the algebraic theory of graph grammars". In: *Proc. 1977 International Conference on Fundamentals of Computation Theory (FCT 1977)*. Ed. by Marek Karpiński. Vol. 56. Lecture Notes in Computer Science. Springer, pp. 245–255. DOI: 10.1007/3-540-08442-8_91.

Ehrig, Hartmut and Hans-Jörg Kreowski (1976). "Parallelism of manipulations in multidimensional information structures". In: *Proc. 5th Symposium on Mathematical Foundations of Computer Science (MFCS 1976)*. Ed. by Antoni Mazurkiewicz. Vol. 45. Lecture Notes in Computer Science. Springer, pp. 284–293. DOI: 10.1007/3-540-07854-1_188.

Farrow, R., K. Kennedy, and L. Zucconi (1976). "Graph grammars and global program data flow analysis". In: *Proc. 17th Annual Symposium on Foundations of Computer Science (SFCS 1976)*. Ed. by E. McCluskey and W. Semon. IEEE, pp. 42–56. DOI: 10.1109/SFCS.1976.17.

Hristakiev, Ivaylo and Detlef Plump (2018). "Checking Graph Programs for Confluence". In: *Software Technologies: Applications and Foundations – STAF 2017 Collocated Workshops, Revised Selected Papers*. Ed. by Martina Seidl and Steffen Zschaler. Vol. 10748. Lecture Notes in Computer Science. Springer, pp. 92–108. DOI: 10.1007/978-3-319-74730-9_8.

Kreowski, Hans-Jörg (1977). "Manipulationen von Graphmanipulationen". PhD thesis. Technische Universität Berlin, Fachbereich Informatik.

References II

Lambers, Leen (2009). "Certifying Rule-Based Models using Graph Transformation". PhD thesis. Technical University of Berlin, Elektrotechnik und Informatik.

Lambers, Leen, Hartmut Ehrig, and Fernando Orejas (2008). "Efficient Conflict Detection in Graph Transformation Systems by Essential Critical Pairs". In: *Proc. Fifth International Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2006)*. Ed. by Roberto Bruni and Dániel Varró. Vol. 211. Electronic Notes in Theoretical Computer Science. Elsevier, pp. 17–26. DOI: 10.1016/j.entcs.2008.04.026.

Lambers, Leen et al. (2018). "Initial Conflicts and Dependencies: Critical Pairs Revisited". In: *Graph Transformation, Specifications, and Nets: In Memory of Hartmut Ehrig*. Ed. by Reiko Heckel and Gabriele Taentzer. Vol. 10800. Lecture Notes in Computer Science. Springer, pp. 105–123. DOI: 10.1007/978-3-319-75396-6_6.

Lambers, Leen et al. (2019). "Exploring Conflict Reasons for Graph Transformation Systems". In: *Proc. 12th International Conference on Graph Transformation (ICGT 2019)*. Ed. by Esther Guerra and Fernando Orejas. Vol. 11629. Lecture Notes in Computer Science. Springer, pp. 75–92. DOI: 10.1007/978-3-030-23611-3_5.

Newman, M. (1942). "On Theories with a Combinatorial Definition of "Equivalence"". In: *Annals of Mathematics* 43.2, pp. 223–243. DOI: 10.2307/1968867.

Plump, Detlef (1998). "Termination of Graph Rewriting is Undecidable". In: *Fundementa Informaticae* 33.2, pp. 201–209. DOI: 10.3233/FI-1998-33204.

— (2005). "Confluence of Graph Transformation Revisited". In: *Processes, Terms and Cycles: Steps on the Road to Infinity: Essays Dedicated to Jan Willem Klop on the Occasion of His 60th Birthday*. Ed. by Aart Middeldorp et al. Vol. 3838. Lecture Notes in Computer Science. Springer, pp. 280–308. DOI: 10.1007/11601548_16.