A Strict Constrained Superposition Calculus for Graphs

R. Echahed, M. Echenim, M. Mhalla and N. Peltier

Univ. Grenoble Alpes, CNRS – Laboratory of Informatics of Grenoble (CAPP)
Partially funded by the French National Research Agency (ANR-22-PETQ-0007)

FoSSaCS – Paris – April 2023

Overview

- **Starting point**: Superposition calculus (a proof procedure for equational reasoning in first-order logic)
- **Our goal:** extend this calculus to handle equations between graphs
- **Roadmap**:
  - Motivation
  - Equational reasoning between first-order terms: the standard superposition calculus (Bachmair and Ganzinger, 94)
  - Superposition for graphs: main issues
  - Theoretical results
  - Future work

Motivation

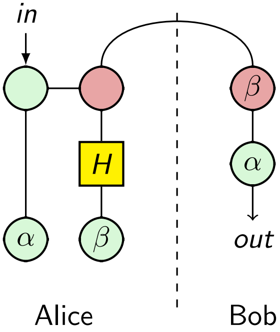
- Graphs are ubiquitous in computer science
- Useful in many applications: to model complex data structures in programming, software and hardware architecture, data bases etc.
- It is often convenient to consider equational theories over such objects

Motivation (2)

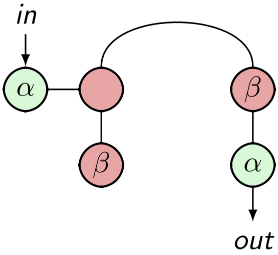
Useful in particular in quantum computing: e.g., ZX calculus

- Describe a quantum transformation (linear map) as a ZX diagram (circuit)
- The semantics may be defined as complex matrices of size $2^{N_{input}+N_{output}}$
- Alternatively, quantum properties can be described by equations between graphs
- Correctness proofs may be performed by proving that two graphs are equal modulo this set of equations

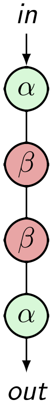
An example: formalization of the quantum teleportation protocol in the ZX calculus



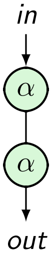
An example: formalization of the quantum teleportation protocol in the ZX calculus



An example: formalization of the quantum teleportation protocol in the ZX calculus

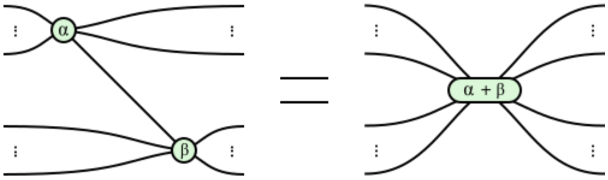


An example: formalization of the quantum teleportation protocol in the ZX calculus



An example: formalization of the quantum teleportation protocol in the ZX calculus

An example of rule



(source: `https://zxcalculus.com/`)

Our Goal

Develop techniques to check the equivalence of two graphs modulo a set of equations

- A *generic* approach
- As general as possible (conditional rules, disjunctions. . .)
- Must be as efficient as possible

Starting Point: Equational Reasoning on Standard Terms

The superposition calculus of Bachmair and Ganziner (94)

- Very efficient and practically successful
- Widely used and thoroughly investigated
- Can the calculus be extended to graphs?

The Superposition Calculus

- = Resolution calculus + Knuth Bendix completion
- Handles set of equational clauses

$$t_1 \approx s_1 \vee \cdots \vee t_n \approx s_n \vee t'_1 \not\approx s'_1 \vee \cdots \vee t'_m \not\approx s'_m$$

  with $n \geq 0$, $m \geq 0$, t_i, s_i, t'_i, s'_i are first-order terms (with variables)
- A set of inference rules deducing new clauses from existing ones
- Very restrictive application conditions, parameterized by an order on terms and literals
- Reduction order = well-founded order closed under embedding and substitution, containing the subterm relation
- A very general criterion to delete redundant clauses

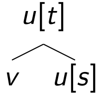
An Example of Rule - Positive Superposition

$$\frac{t \approx s \vee C \qquad u[t'] \approx v \vee D}{(u[s] \approx v \vee C \vee D)\sigma}$$

if:

- σ is the most general unifier of t' and t
- $t\sigma \not< s\sigma$
- $(u[t'] \approx v)\sigma$ is maximal in $(u[t'] \approx v \vee D)\sigma$
- $(t \approx s)\sigma$ is maximal in $(t \approx s \vee C)\sigma$

Intuition: compute critical pairs of rules $t \rightarrow s$ and $u \rightarrow v$



Redundancy Criterion

Redundant Clause

A clause C is redundant in S if for every ground instance $C\sigma$ of C, there exist ground instances $D_1\theta_1, \ldots, D_n\theta_n$ (with $n \geq 0$) of clauses in S such that:

- $C\sigma$ is a logical consequence of $D_1\theta_1, \ldots, D_n\theta_n$
- $C\sigma$ is (strictly) greater than $D_1\theta_1, \ldots, D_n\theta_n$

For instance, all tautological ($=$ valid) clauses are redundant

Properties of the Superposition Calculus

- Sound
- Refutationally complete
- Very efficient in practice
- Can even be used as a decision procedure for several fragments
- Numerous extensions

Can we do the same for graphs?

Numerous issues, regarding completeness:

- Can we reason modulo isomorphism?
- Can we use the same redundancy criterion ?
- What about reduction orders?

Lifting Superposition to Graphs: First Completeness Issue

Reasoning up to isomorphism is not always sufficient:

The equation can be considered as trivial since the two graphs are isomorphic

But it contradicts the following disequation:

- From the standpoint of Superposition: the calculus is incomplete: no clauses can be derived if graphs are taken up to isomorphism
- From the standpoint of rewriting: confluence is hard to establish for graph rewrite rules
  - The critical pair lemma is not true
  - Confluence is not decidable for terminating systems (Plump 05)

Lifting Superposition to Graphs: A Unsatisfactory Solution

A trivial solution: name all the nodes

The equation is not tautological anymore... but redundancy deletion becomes very weak

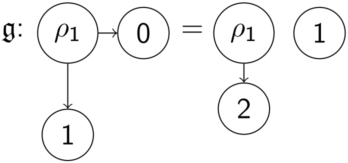
How to Overcome the Confluence Issue?

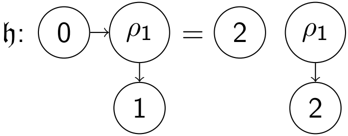
Use graphs with interface

- Interface = a sequence of distinguished *named* nodes (the roots)
- Allow one to connect the graph to the outside world
- The other (non root) nodes can be renamed arbitrarily. . .
- . . . but cannot be linked to the outside of the graph
- A trade-off between the flexibility of graph composition and the strength of redundancy deletion

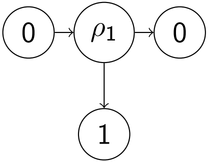
Lifting Superposition to Graphs: Redundancy

More general inference rules are required:

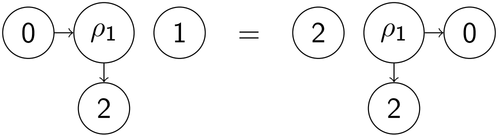




We can "merge" $\mathfrak{g}$ and $\mathfrak{i}$ as follows:



We deduce:

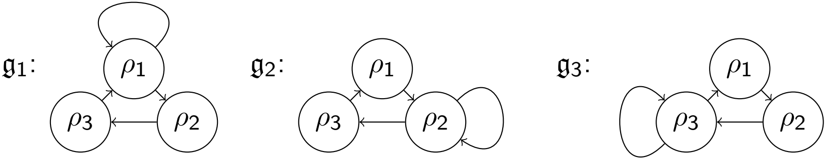


Lifting Superposition to Graphs: Redundancy (2)

- The previous example shows that the conclusion of a rule can be *strictly greater* than both premises
- Not compatible with the usual redundancy criterion:
  such a conclusion is always redundant (in the usual sense)
  hence the inference will be blocked

Lifting Superposition to Graphs: Tautologies

The calculus is incomplete if tautologies are deleted. Consider the graphs $\mathfrak{g}_1, \mathfrak{g}_2$ and $\mathfrak{g}_3$ with roots (ρ_1, ρ_2, ρ_3):



- Let $\dot{\mathfrak{g}}_i$ be a graph obtained from $\mathfrak{g}_i$ by adding an isolated node
- $S = \{\dot{\mathfrak{g}}_1 \approx \mathfrak{g}_2 \vee \dot{\mathfrak{g}}_2 \approx \mathfrak{g}_3 \vee \dot{\mathfrak{g}}_3 \approx \mathfrak{g}_1, \dot{\mathfrak{g}}_1 \not\approx \mathfrak{g}_2 \vee \dot{\mathfrak{g}}_2 \not\approx \mathfrak{g}_3 \vee \dot{\mathfrak{g}}_3 \not\approx \mathfrak{g}_1\}$
- S is *unsatisfiable* but *cannot be refuted* if tautologies are deleted

How to Overcome the Redundancy Issue?

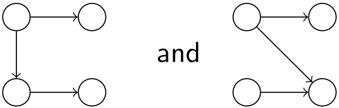
Use a much more restricted redundancy criterion

- Based on a carefully designed set of simplification rules
  - Demodulation (equational simplification)
  - Subsumption
  - Deletion of trivial equations or disequations (modulo isomorphism)
- A clause is redundant if it can be reduced to $\top$ using the set of simplification rules
- Tautology deletion is possible only in some very specific cases (Horn clauses)

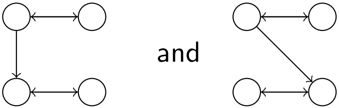
Lifting Superposition to Graphs: Order Issue

No reduction order that is total on ground graphs exists

- Consider the graphs:



- These two graphs are distinct (non-isomorphic) hence one of them must be strictly greater than the other
- But if we add the same two edges in each graph, we get two isomorphic graphs (contradicting the closure under embedding requirement):



How to Overcome the Order Issue?

Use orders that are not total on ground graphs (e.g.: number of nodes)

- Not problematic for defining the calculus (reduction orders are not complete anyway for non ground terms)
- However, total reduction orders are essential for the completeness proof
- Completeness is usually ensured by constructing a model of saturated sets of clauses (not containing $\square$)
- The model is described as a convergent set of equations
- Termination is a ensured by orienting the rules: $t \approx s$ yields $t \rightarrow s$ if $t > s$
- Confluence stems for the fact that the considered set is saturated

How to Overcome the Order Issue? (2)

Adapt the completeness proof

- If no total reduction order exists, then some equations cannot be oriented anymore ($t \approx s$ yields both $t \rightarrow s$ and $s \rightarrow t$)
- The obtained rewrite system is not terminating
- Confluence is more difficult to establish (local confluence is not enough)
- Our solution: a new confluence criterion, based on (an extension of) subcommutative relations

Results

- A new class of graphs for which a sound and complete calculus can be defined
- Sufficiently expressive to encode ZX diagrams (or similar circuits, with distinguished input/output edges)

Results (2)

The calculus is defined in two steps:

- Defined first for non-interpreted (ground) labels
- In a second step, the calculus is extended into a constrained-based calculus, handling labels with variables, interpreted in any decidable theory (e.g., graph with arithmetic labels on vertices)
- Extract the conditions on the labels that make the application of inference rule possible and add them to the constraints of the conclusion
- Adapt all simplification rules to handle constraints

Main Theorems

Soundness

For all clause sets S, for all constrained clauses $[C \mid \mathcal{X}]$ deducible from S, for all substitutions σ such that $\mathcal{X}\sigma$ is true (in the label theory), $C\sigma$ is a logical consequence of S.

Completeness

If S is unsatisfiable and saturated under all inference rules (w.r.t. the redundancy criterion) then there exists a set of constrained clauses $\{[\Box \mid \mathcal{X}_i] \mid i \in I\}$ such that $\bigwedge_{i \in I} \neg \mathcal{X}_i$ is unsatisfiable (in the label theory).
If, moreover, the label theory is compact, then I is finite, and unsatisfiability is thus semi-decidable.

An Example of Rule

The Graph Positive Superposition rule

$$\frac{[\mathfrak{g}_1 \approx \mathfrak{h}_1 \vee C_1 \mid \phi_1] \quad [\mathfrak{g}_2 \approx \mathfrak{h}_2 \vee C_2 \mid \phi_2]}{[\mathfrak{i}\{\mathfrak{g}_1 \leftarrow \mathfrak{h}_1\} \approx \mathfrak{i}\{\mathfrak{g}_2 \leftarrow \mathfrak{h}_2\} \vee C_1 \vee C_2 \mid \phi_1 \wedge \phi_2 \wedge \psi]}$$

where:

1. $\mathfrak{i}$ is a "merge" of $\mathfrak{g}_1$ and $\mathfrak{g}_2$ with constraint ψ, and $\mathfrak{g}_1$ and $\mathfrak{g}_2$ are not "disjoint";
2. $\mathfrak{g}_i \approx \mathfrak{h}_i$ is maximal in $[\mathfrak{g}_i \approx \mathfrak{h}_i \vee C_i \mid \phi_1 \wedge \phi_2 \wedge \psi]$ (for all $i = 1, 2$);
3. $\mathfrak{g}_i$ is not strictly lower than $\mathfrak{h}_i$ (for all $i = 1, 2$) taking into account the constraints $\phi_1 \wedge \phi_2 \wedge \psi$).

Future Work

- Implementation
- How to prune the search space?
- How to represent huge sets of graphs efficiently?
- Graph variables
- Termination issues
- Add new rules to enable tautology deletion