CSE599d: Advanced Query Processing

Lecture 7: Extensible Query Optimizers

Dan Suciu

University of Washington

Motivation

Dynamic programming is efficient, but handles only joins.

Difficult to add more complex operators (outer/anti joins, group-by).

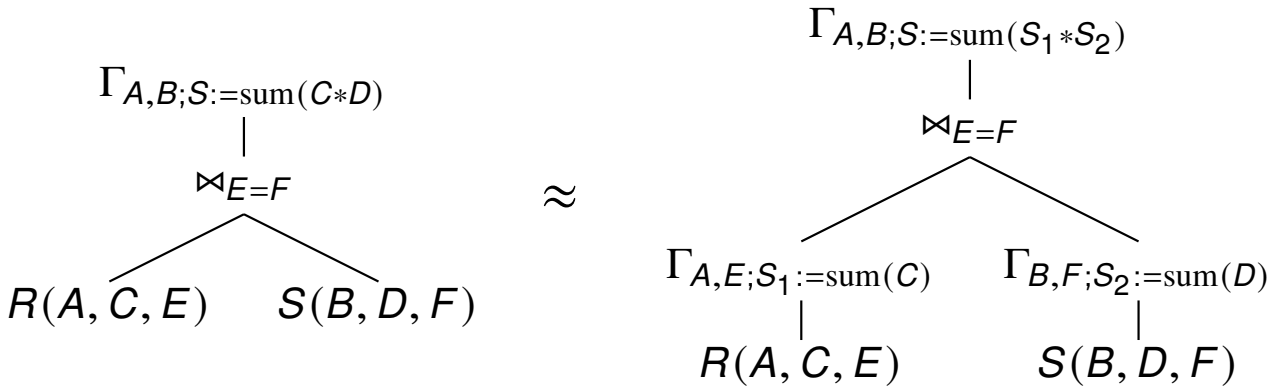
Extensible Query Optimizer: based on a set of identities called logical transformation rules, which can be extended with new operators and new identities.

Examples of Identities[1]

Commutativity:

Associativity:

Group-by:



[1] Called Logical Transformation Rules in Volcano/Cascades.

Overview

- The optimization problem becomes an instance of the word problem
- There are two techniques for the word problem: TRS and E-graphs.
- Volcano and Cascades (an extension of Volcano) are essentially E-graphs on steroids.
- This lecture: review of the Word Problem, TRS, E-Graphs.

The Word Problem

Based on [Baader and Nipkow, 1999].

Σ-Algebras

A signature is $\Sigma = \{f_1, f_2, \ldots, f_m\}$
each function symbol f_i and has an arity $n_i \geq 0$.

Σ-Algebras

A signature is $\Sigma = \{f_1, f_2, \ldots, f_m\}$
each function symbol f_i and has an arity $n_i \geq 0$.

A Σ-algebra is $\boldsymbol{A} = (A, f_1^A, f_2^A, \ldots, f_m^A)$ where $f_i^A : A^{n_i} \rightarrow A$.

Σ-Algebras

A signature is $\Sigma = \{f_1, f_2, \ldots, f_m\}$
each function symbol f_i and has an arity $n_i \geq 0$.

A Σ-algebra is $\boldsymbol{A} = (A, f_1^A, f_2^A, \ldots, f_m^A)$ where $f_i^A : A^{n_i} \to A$.

Examples:

- $\Sigma = \{f, g\}$ with arities $1, 2$.
  Algebra $(\mathbb{N}, f^N, g^N)$ where $f^N(x) = x + 1$ and $g^N(x, y) = x \cdot y$.

Σ-Algebras

A signature is $\Sigma = \{f_1, f_2, \ldots, f_m\}$
each function symbol f_i and has an arity $n_i \geq 0$.

A Σ-algebra is $\boldsymbol{A} = (A, f_1^A, f_2^A, \ldots, f_m^A)$ where $f_i^A : A^{n_i} \to A$.

Examples:

- $\Sigma = \{f, g\}$ with arities $1, 2$.
  Algebra $(\mathbb{N}, f^N, g^N)$ where $f^N(x) = x + 1$ and $g^N(x, y) = x \cdot y$.

- $\Sigma = \{+, \times, 0, 1\}$
  Algebra $\mathbb{N} = (\{0, 1, 2, \ldots\}, +, \times, 0, 1)$
  Another algebra $\mathbb{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$

The Term Algebra

Fix an infinite set of variables $X = \{x_1, x_2, \ldots\}$.

A Σ-term is an expression built from Σ and X.

$$\boxed{T(\Sigma, X) = \text{the set of } \Sigma \text{ terms}}$$

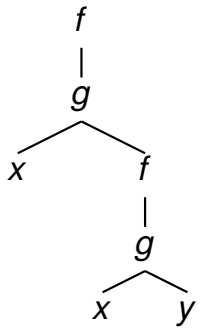
The Term Algebra

Fix an infinite set of variables $X = \{x_1, x_2, \ldots\}$.

A Σ-term is an expression built from Σ and X.

$$\boxed{T(\Sigma, X) = \text{the set of } \Sigma \text{ terms}}$$

$f(g(x, f(g(x, y)))) =$



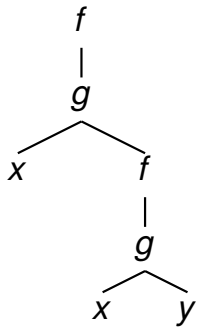
The Term Algebra

Fix an infinite set of variables $X = \{x_1, x_2, \ldots\}$.

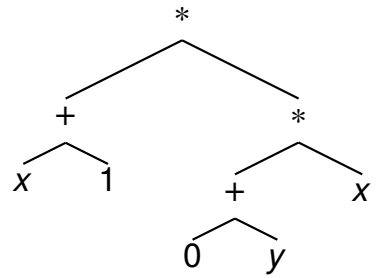
A Σ-term is an expression built from Σ and X.

$$\boxed{T(\Sigma, X) = \text{the set of } \Sigma \text{ terms}}$$

$f(g(x, f(g(x, y)))) =$



$(x + 1) * ((0 + y) * x) =$



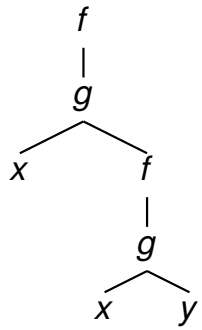
The Term Algebra

Fix an infinite set of variables $X = \{x_1, x_2, \ldots\}$.

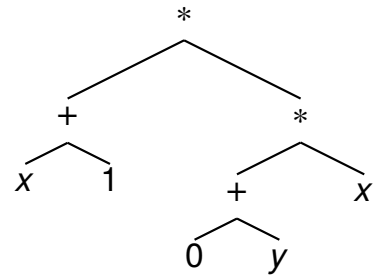
A Σ-term is an expression built from Σ and X.

$$\boxed{T(\Sigma, X) = \text{the set of } \Sigma \text{ terms}}$$

$f(g(x, f(g(x, y)))) =$



$(x + 1) * ((0 + y) * x) =$



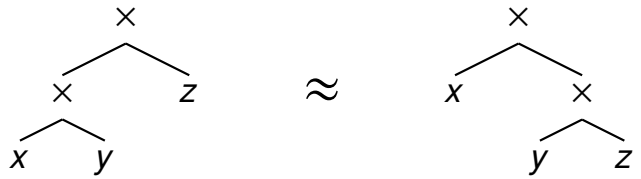
$\boxed{T(\Sigma, X) \text{ is a } \Sigma \text{ algebra}}$

what are the operations $f, g, \ldots$??

Identities

An identity is a pair of terms (ℓ, r). We write $\ell \approx r$.

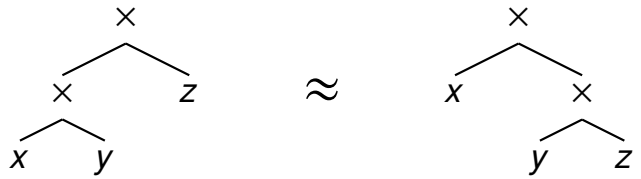
ℓ = the left-hand-side
r = the right-hand-side



Identities

An identity is a pair of terms (ℓ, r). We write $\ell \approx r$.

ℓ = the left-hand-side
r = the right-hand-side

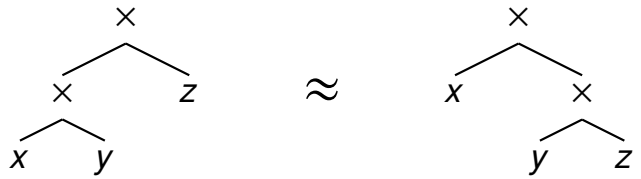


What does $\ell \approx r$ mean?

Identities

An identity is a pair of terms (ℓ, r). We write $\ell \approx r$.

ℓ = the left-hand-side
r = the right-hand-side



What does $\ell \approx r$ mean? Fix an algebra $\boldsymbol{A}$:

Identities

An identity is a pair of terms (ℓ, r). We write $\ell \approx r$.

ℓ = the left-hand-side
r = the right-hand-side

What does $\ell \approx r$ mean? Fix an algebra $\boldsymbol{A}$:

A substitution is a function $\sigma : X \to A$, where $\boldsymbol{A}$ is a Σ-algebra.

$\boldsymbol{A}$ satisfies the identity $\ell \approx r$ if $\sigma(\ell) = \sigma(r)$ for all σ. Notation: $\boxed{\boldsymbol{A} \models \ell \approx r}$

Semantic Consequence

Fix a set of identities E.

An identity $s \approx t$ is a semantic consequence of E if:

For every Σ-algebra $\boldsymbol{A}$: $\boxed{\text{if } \boldsymbol{A} \models E \text{ then } \boldsymbol{A} \models s \approx t}$

We write $\approx_E$ for the semantic consequences of E.

It turns out that we can describe $\approx_E$ in terms of reductions.

The Reduction Relation

The reduction relation $\boxed{s \to_E t}$ holds if
there exists $(\ell, r) \in E$, substitution $\sigma : X \to T(\Sigma, X)$, context [2] $C[\ ]$ s.t.:

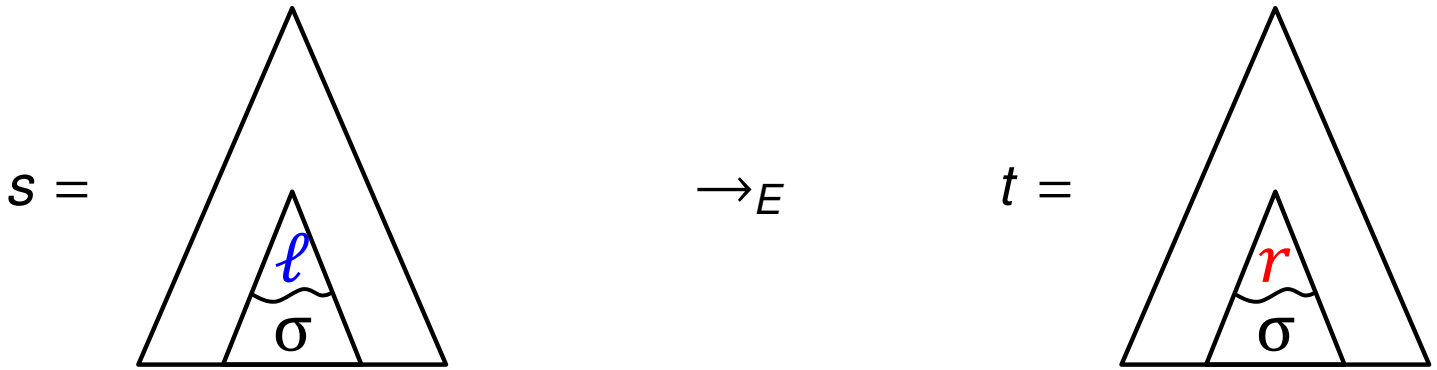
$$\boxed{s = C[\sigma(\ell)] \text{ and } t = C[\sigma(r)]}$$

[2] A context is a "term with a hole".

The Reduction Relation

The reduction relation $\boxed{s \to_E t}$ holds if
there exists $(\ell, r) \in E$, substitution $\sigma : X \to T(\Sigma, X)$, context [2] $C[\ ]$ s.t.:

$$\boxed{s = C[\sigma(\ell)] \text{ and } t = C[\sigma(r)]}$$



[2] A context is a "term with a hole".

The Reduction Relation

We extend $\rightarrow_E$ as follows:

$\rightarrow_E^*$ transitive and reflexive closure of $\rightarrow_E$

$\leftarrow_E$ symmetric closure of $\rightarrow_E$

$\leftrightarrow_E^*$ transitive, reflexive and symmetric closure of $\rightarrow_E$

We say that $s \leftrightarrow_E^* t$ is a syntactic consequence of E.

Theorem [Birkoff] $\approx_E$ is the same as $\leftrightarrow_E^*$

The Word Problem

Fix Σ, E.

The Word Problem: given s, t, decide whether $s \approx_E t$

Decidability depends on Σ, E.

Examples of Decidable Word Problems

Semigroups $\Sigma = \{\cdot\}$

$$(xy)z \approx x(yz)$$

Examples of Decidable Word Problems

Semigroups $\Sigma = \{\cdot\}$

$$(xy)z \approx x(yz)$$

Also:

- Commutative Semigroups
  $xy \approx yx$
- Monoids $ex \approx xe \approx x$
- Groups $x^{-1}x \approx xx^{-1} \approx e$.

Examples of Decidable Word Problems

Semigroups $\Sigma = \{\cdot\}$

$$(xy)z \approx x(yz)$$

Also:

- Commutative Semigroups
  $xy \approx yx$
- Monoids $ex \approx xe \approx x$
- Groups $x^{-1}x \approx xx^{-1} \approx e$.

Lattices $\Sigma = \{\vee, \wedge\}$

$\wedge, \vee$ are associative, commutative, and:

$$x \vee (x \wedge y) \approx x$$
$$x \wedge (x \vee y) \approx x$$

Also:

- Distributive lattices
- Boolean algebras

Examples of Undecidable Word Problems

Semigroups [Matijasevič] $\Sigma = \{\cdot, a, b\}$

$$(xy)z \approx x(yz)$$
$$aba^2b^2 \approx b^2a^2ba$$
$$a^2bab^2a \approx b^2a^3ba$$
$$aba^3b^2 \approx ab^2aba^2$$
$$b^3a^2b^2a^2ba \approx b^3a^2b^2a^4$$
$$a^4b^2a^2ba \approx b^2a^4$$

Combinatory Logic $\Sigma = \{\cdot, S, K\}$

$$((S \cdot x) \cdot y) \cdot z \approx (x \cdot z) \cdot (y \cdot z)$$
$$(K \cdot x) \cdot y \approx x$$

A Special Case

Theorem when all terms in E are grounded (meaning: no variables), then the ground word problem[3] for E is decidable.

[3] Meaning: s, t are also grounded.

QO v.s. the Word Problem

Query Optimization v.s. the Word Problem

Query optimization: given a set of logical transformations E (i.e. identities), and a query Q, find an "optimal" query Q' such that $Q \approx_E Q'$.

This is basically a word problem.

However, there are some rough spots in modeling QO as a word problem.

Relational Algebra v.s. a Σ-algebra

The relational algebra is a Σ-algebra, where $\Sigma = \{\sigma, \Pi, \bowtie, \cup, -\}$, however:

- Some operations are undefined: $R \cup S$ when R, S have different schemas.
- Some operations have parameters: $\sigma_p(R)$ where p is some predicate.
- Template identities: one $R \bowtie S \approx S \bowtie R$ for each schema of R, S.

Could fix with a multi-sorted algebra. In practice, just ignore these rough edges.

E is Always Incomplete!

Query equivalence, $Q \equiv Q'$, means: $Q(D) = Q'(D)$ for any database D.

Fact There is no r.e.[4] E for which $\approx_E$ coincides with $\equiv$.

[4] Recursively enumerable, e.g. templates.

E is Always Incomplete!

Query equivalence, $Q \equiv Q'$, means: $Q(D) = Q'(D)$ for any database D.

Fact There is no r.e.[4] E for which $\approx_E$ coincides with $\equiv$.

Proof Recall that checking $Q \equiv Q'$ is undecidable.

[4] Recursively enumerable, e.g. templates.

E is Always Incomplete!

Query equivalence, $Q \equiv Q'$, means: $Q(D) = Q'(D)$ for any database D.

Fact There is no r.e.[4] E for which $\approx_E$ coincides with $\equiv$.

Proof Recall that checking $Q \equiv Q'$ is undecidable.

Assume $\approx_E$ and $\equiv$ are the same. Then we can check $Q \equiv Q'$:

- Enumerate all pairs (Q_1, Q_2) such that $Q_1 \approx_E Q_2$ how??
- Enumerate all pairs (Q_1, Q_2) such that $Q_1 \not\equiv Q_2$ how??
- Watch whether (Q, Q') appears in the first or second list.

[4] Recursively enumerable, e.g. templates.

Term Rewriting Systems

Two Approaches to the Word Problem

- Term Rewriting Systems (TRS)
- Equality Saturation Graphs (E-Graphs)

Term Rewriting Systems

A Term Rewriting Systems (TRS) is a set of oriented identities E, written $\ell \rightarrow r$.

Goal: study the properties of $\overset{*}{\rightarrow}_E$.

Note: TRS require $\text{Vars}(\ell) \supseteq \text{Vars}(r)$ and ℓ is not a variable

Solving the Word Problem via TRS

A Naive Approach to check $s \approx_E t$:

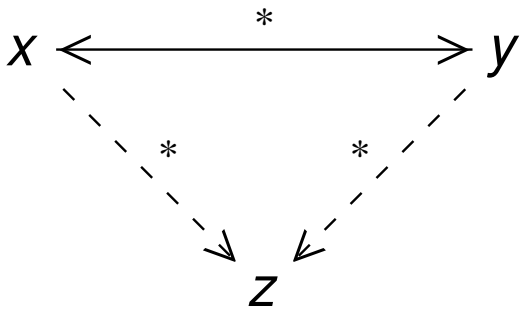
Perform reductions $s \to s_1 \to s_2 \to \cdots$ If t shows up, then $s \equiv_E t$.

Problems with the naive approach:

- Infinite reductions. E.g. commutativity $a + b \to b + a \to a + b \to \cdots$
- We may need to go in both directions $s \to s_1 \leftarrow s_2 \leftarrow s_3 \to \cdots$.

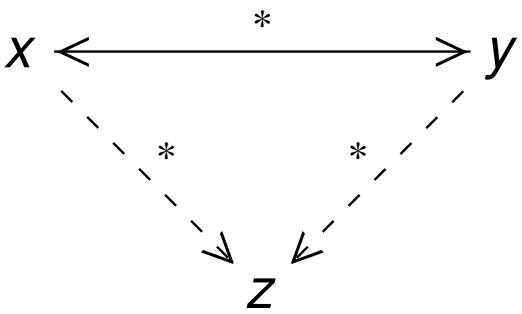
Desirable Properties of a TRS

Church-Rosser:

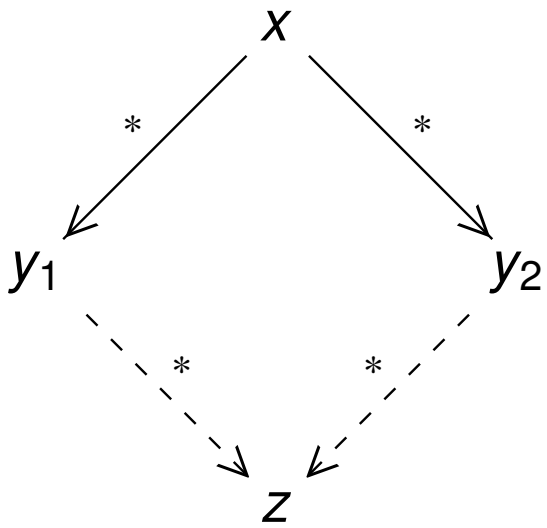


Desirable Properties of a TRS

Church-Rosser:

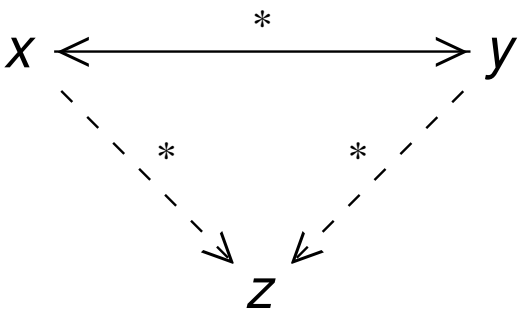


Confluent:

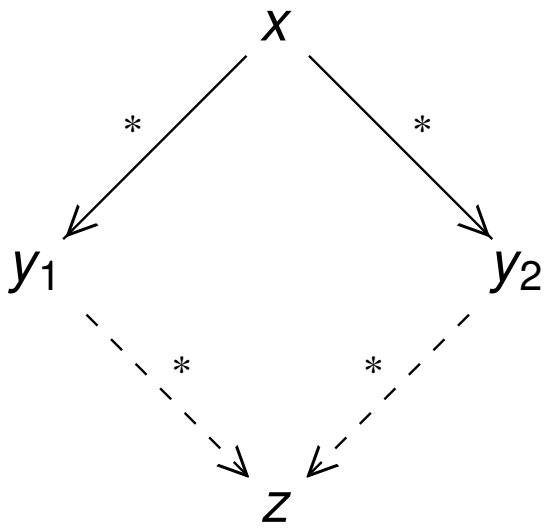


Desirable Properties of a TRS

Church-Rosser:



Confluent:



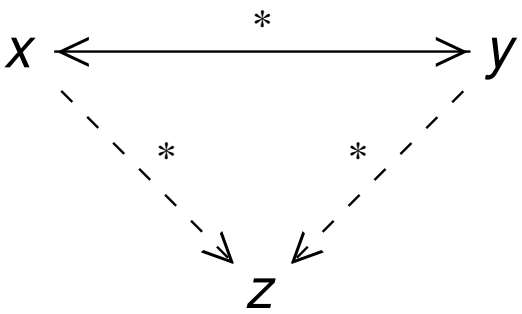
Terminating:

No infinite sequence:

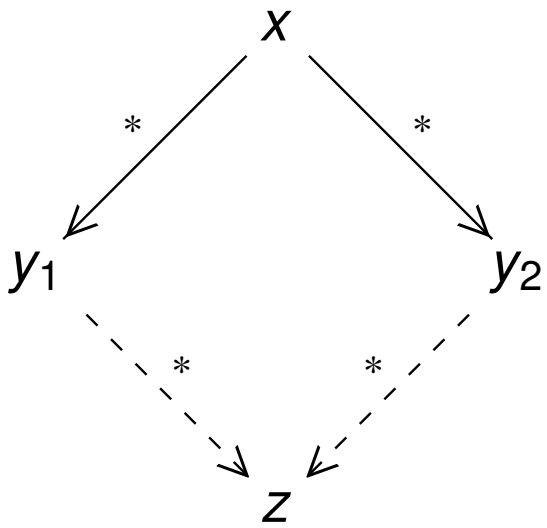
$$x_0 \longrightarrow x_1 \longrightarrow x_2 \longrightarrow \cdots$$

Desirable Properties of a TRS

Church-Rosser:



Confluent:



Terminating:

No infinite sequence:

$x_0 \longrightarrow x_1 \longrightarrow x_2 \longrightarrow \cdots$

- $\longrightarrow$ is Church-Rosser iff it is confluent.
- If $\longrightarrow$ is confluent and terminating then $\approx$ (which is $\overset{*}{\leftrightarrow}$) is decidable. because each x has a unique normal form x_0: $x \overset{*}{\longrightarrow} x_0 \not\rightarrow$

Discussion

Key assumption in TRS: identities are directional. $\ell \approx r$ not same as $r \approx \ell$.

Problems in TRS: non-terminating reductions.

Rich literature on sufficient conditions to ensure termination.
Still can't handle commutativity.

E-Graphs were introduced as a caching mechanism to avoid repeated terms.

E-Graphs

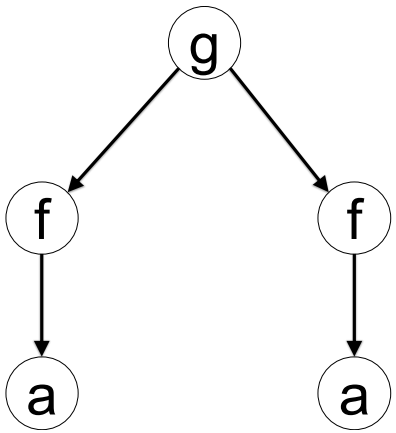
Overview

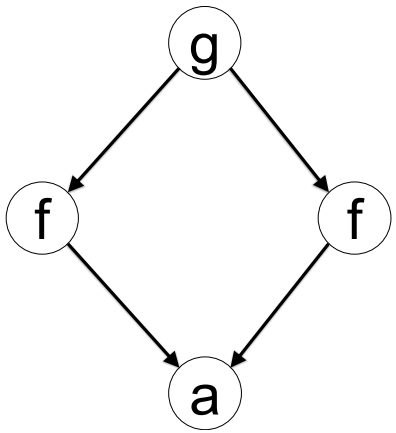
E-graphs are keeping all intermediate steps of a rewrite sequence $s \to s_1 \to s_2 \to \cdots$ in a compact data structure.

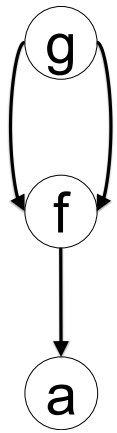
This part is based on the book [Baader and Nipkow, 1999] and the egg paper [Willsey et al., 2021].

Term Graph

Modify tree to a DAG by sharing common subexpressions: Term Graph







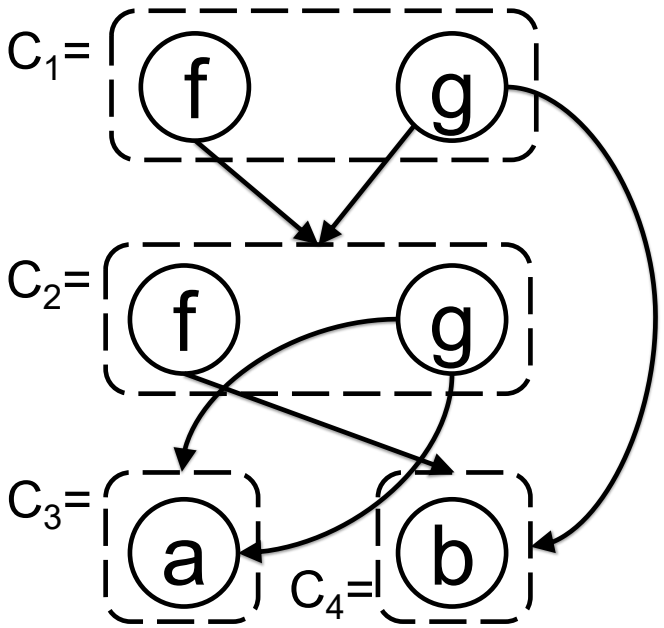
Each of these term graphs represent the term $g(f(a), f(a))$.

E-Graph

An E-graph G consists of:

- A set of E-classes $c_1, c_2, \ldots,$
- $\ldots$ where each E-class c is a set of E-nodes $c = \{v_1, v_2, \ldots\}$,
- $\ldots$ and each e-node is a pair $(f, (c_{i_1}, \ldots, c_{i_k}))$ where $f \in \Sigma$, $c_{i_1}, \ldots, c_{i_k}$ are E-classes, and $k =$ the arity of f.

E-Graphs



Each class c represents a set of terms:

$\text{terms}(c_3) = a,$

$\text{terms}(c_4) = b$

$\text{terms}(c_2) = \{f(b), g(a,a)\}$

$$\text{terms}(c_1) = \{f(f(b)), f(g(a,a)), g(f(b),b), g(g(a,a),b)\}$$

$\text{terms}(G)$ = the set of all terms represented by all E-classes.

Congruence Closure

An E-graph defines the following relation on terms(G):

$$s \sim_G t \text{ if exists E-class } c \text{ such that } s, t \in \text{terms}(c)$$

Congruence Closure

An E-graph defines the following relation on terms(G):

$$\boxed{s \sim_G t \text{ if exists E-class } c \text{ such that } s, t \in \text{terms}(c)}$$

We say that G is closed under congruence if $\sim_G$ is a congruence relation on terms(G), meaning:

- $\sim_G$ is an equivalence relation, and
- if $f(s_1, \ldots, s_n), f(t_1, \ldots, t_n) \in \text{terms}(G)$ and $s_1 \sim_G t_1, s_2 \sim_G t_2, \ldots$ then $f(s_1, \ldots, s_n) \sim_G f(t_1, \ldots, t_n)$.

Congruence Closure

An E-graph defines the following relation on terms(G):

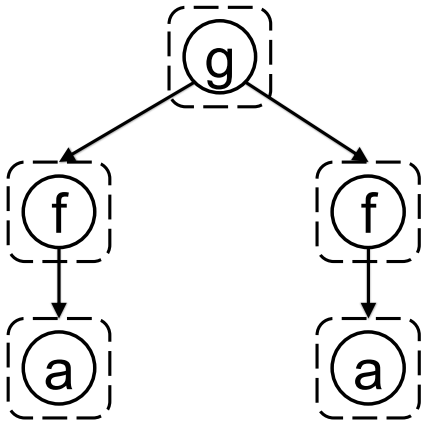
$$\boxed{s \sim_G t \text{ if exists E-class } c \text{ such that } s, t \in \text{terms}(c)}$$

We say that G is closed under congruence if $\sim_G$ is a congruence relation on terms(G), meaning:

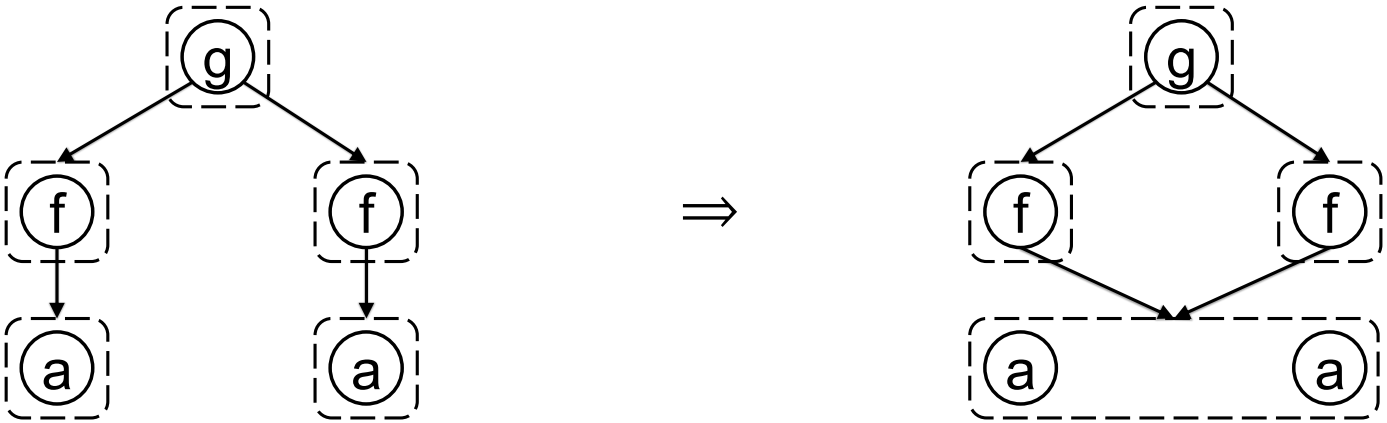
- $\sim_G$ is an equivalence relation, and
- if $f(s_1, \ldots, s_n), f(t_1, \ldots, t_n) \in \text{terms}(G)$ and $s_1 \sim_G t_1, s_2 \sim_G t_2, \ldots$ then $f(s_1, \ldots, s_n) \sim_G f(t_1, \ldots, t_n)$.

Congruence closure algorithm: modifies G to ensure that $\sim_G$ is a congruence.

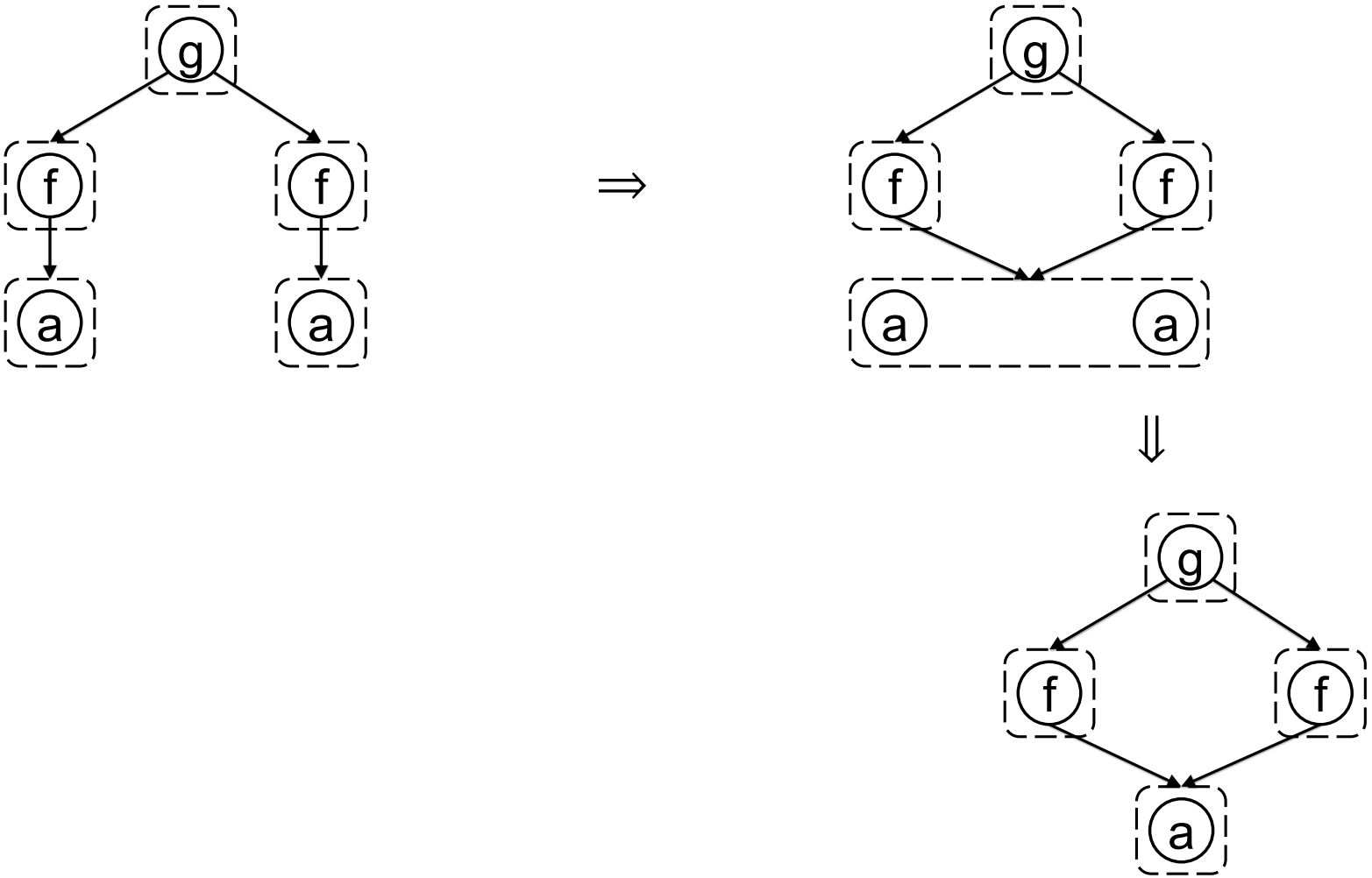
Example



Example

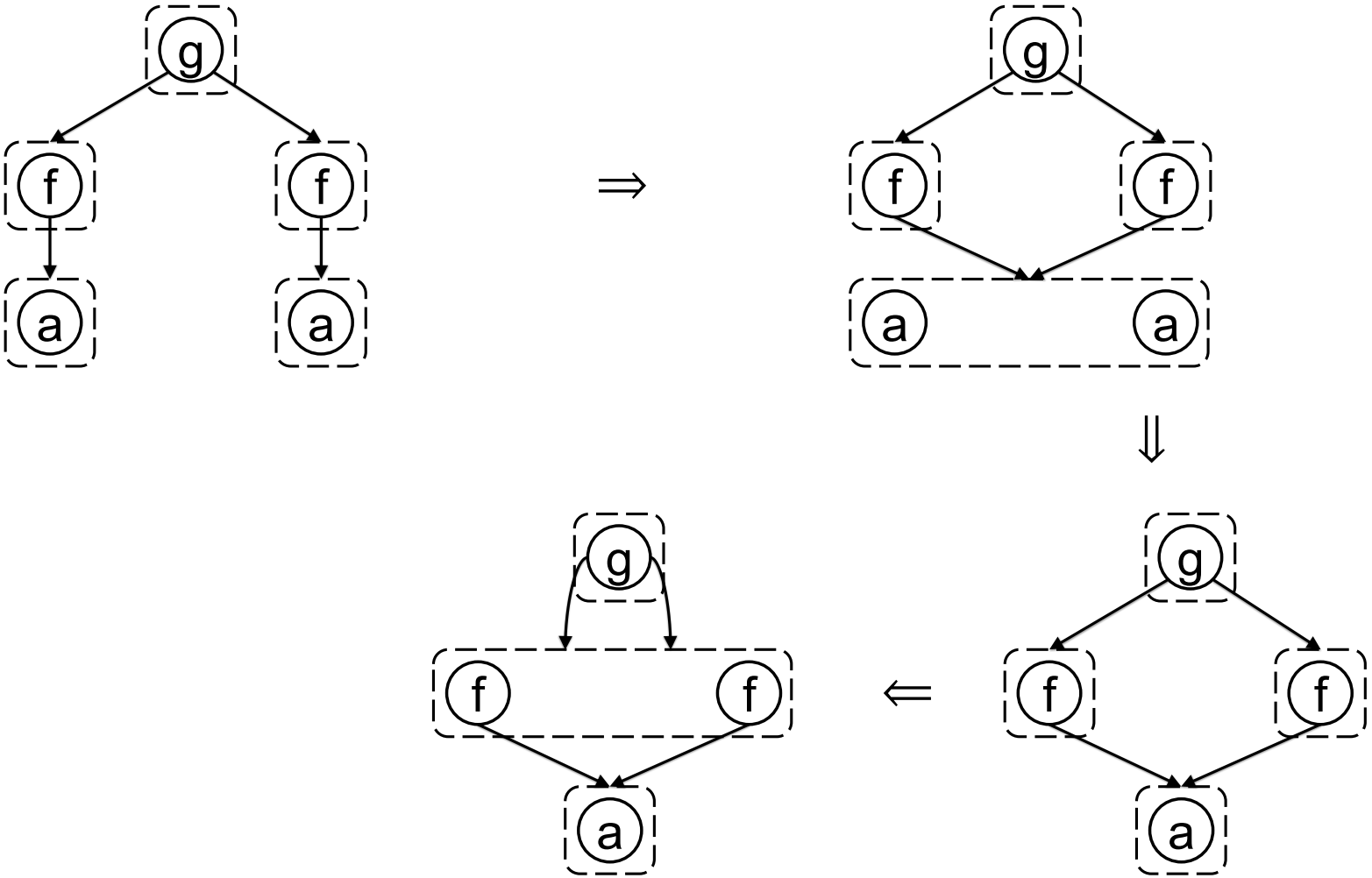


Example



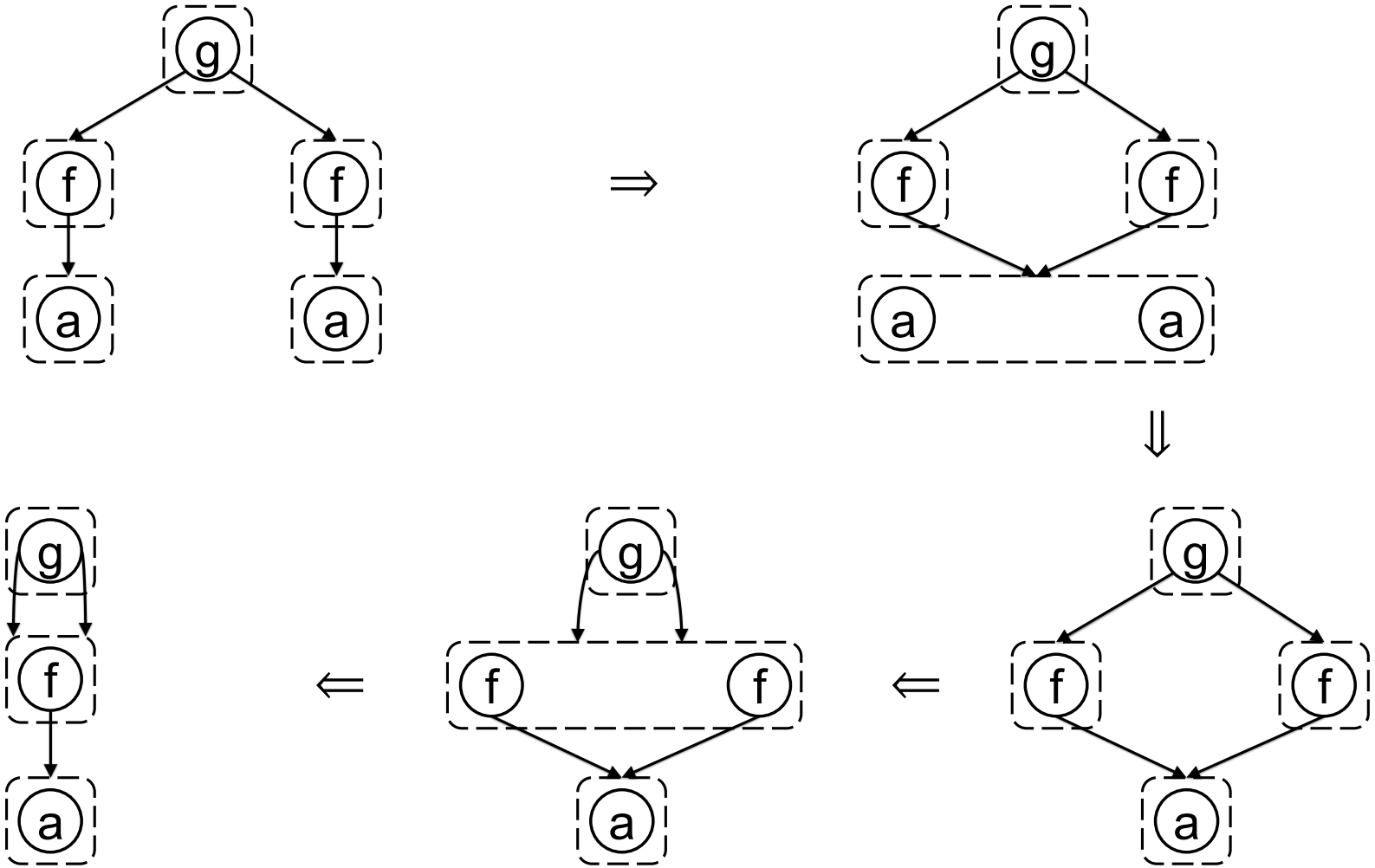
Two E-nodes with the same label and same children are the same E-node.

Example



Two E-nodes with the same label and same children are the same E-node.

Example



Two E-nodes with the same label and same children are the same E-node.

The Equality Saturation Algorithm for the Word Problem

To check $\boxed{s \approx_E t}$, start with the E-graph encoding s and t.

- Find $(\ell, r) \in E$ such that ℓ "matches" the E-graph.
  Meaning: there exists substitution σ, s.t. $\sigma(\ell) \in \text{terms}(G)$.
  Notice: identities are directional.

The Equality Saturation Algorithm for the Word Problem

To check $\boxed{s \approx_E t}$, start with the E-graph encoding s and t.

- Find $(\ell, r) \in E$ such that ℓ "matches" the E-graph.
  Meaning: there exists substitution σ, s.t. $\sigma(\ell) \in \text{terms}(G)$.
  Notice: identities are directional.

- Add $\sigma(r)$ to G (if not already present).

The Equality Saturation Algorithm for the Word Problem

To check $\boxed{s \approx_E t}$, start with the E-graph encoding s and t.

- Find $(\ell, r) \in E$ such that ℓ "matches" the E-graph.
  Meaning: there exists substitution σ, s.t. $\sigma(\ell) \in \text{terms}(G)$.
  Notice: identities are directional.
- Add $\sigma(r)$ to G (if not already present).
- Perform congruence closure
- Repeat.

The Equality Saturation Algorithm for the Word Problem

To check $\boxed{s \approx_E t}$, start with the E-graph encoding s and t.

- Find $(\ell, r) \in E$ such that ℓ "matches" the E-graph.
  Meaning: there exists substitution σ, s.t. $\sigma(\ell) \in \text{terms}(G)$.
  Notice: identities are directional.
- Add $\sigma(r)$ to G (if not already present).
- Perform congruence closure
- Repeat.

When (and if!) it terminates, check $\boxed{s \sim_G t}$.

Example from [Baader and Nipkow, 1999]

$E = \boxed{g(x) \approx f(f(x))}$

Check if $\boxed{g(f(a)) \approx f(g(a))}$

Example from [Baader and Nipkow, 1999]

$E = \boxed{g(x) \approx f(f(x))}$

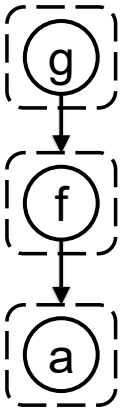
Check if $\boxed{g(f(a)) \approx f(g(a))}$
Need to close E under symmetry

Example from [Baader and Nipkow, 1999]

$E = \boxed{\begin{array}{l} g(x) \approx f(f(x)) \\ f(f(x)) \approx g(x) \end{array}}$

Check if $\boxed{g(f(a)) \approx f(g(a))}$

Need to close E under symmetry

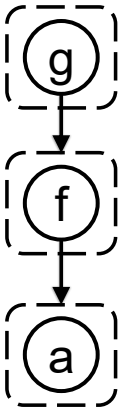


Example from [Baader and Nipkow, 1999]

$$E = \boxed{\begin{array}{l} g(x) \approx f(f(x)) \\ f(f(x)) \approx g(x) \end{array}}$$

Check if $\boxed{g(f(a)) \approx f(g(a))}$

Need to close E under symmetry



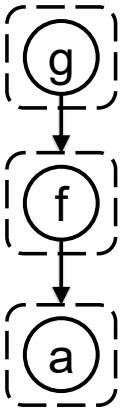
$$g(x) \approx f(f(x))$$

Example from [Baader and Nipkow, 1999]

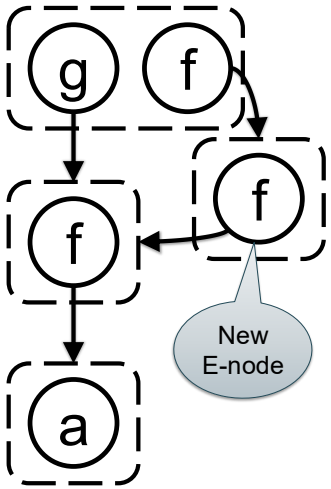
$E = \boxed{\begin{array}{l} g(x) \approx f(f(x)) \\ f(f(x)) \approx g(x) \end{array}}$

Check if $\boxed{g(f(a)) \approx f(g(a))}$

Need to close E under symmetry



$g(x) \approx f(f(x))$

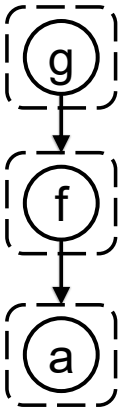


Example from [Baader and Nipkow, 1999]

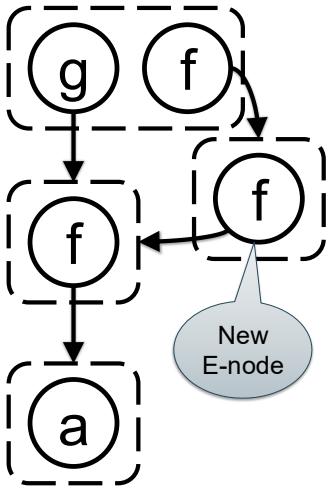
$E = \boxed{\begin{array}{l} g(x) \approx f(f(x)) \\ f(f(x)) \approx g(x) \end{array}}$

Check if $\boxed{g(f(a)) \approx f(g(a))}$

Need to close E under symmetry



$g(x) \approx f(f(x))$



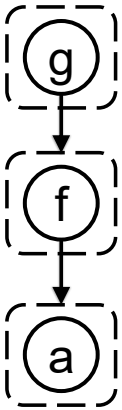
$f(f(x)) \approx g(x)$

Example from [Baader and Nipkow, 1999]

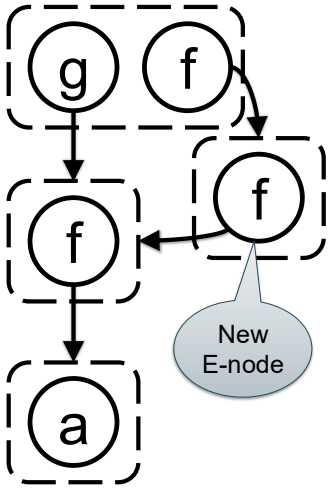
$$E = \boxed{\begin{array}{l} g(x) \approx f(f(x)) \\ f(f(x)) \approx g(x) \end{array}}$$

Check if $\boxed{g(f(a)) \approx f(g(a))}$

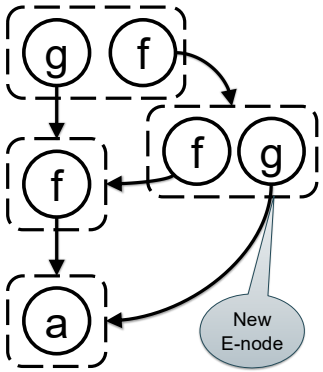
Need to close E under symmetry



$g(x) \approx f(f(x))$

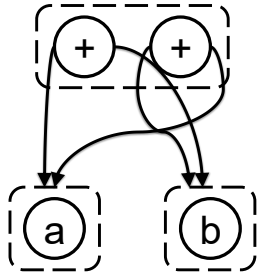


$f(f(x)) \approx g(x)$



Discussion

E-graphs can avoid some non-terminating reductions:,
E.g. commutativity:



Non-termination can still happen, e.g. $g(x) \approx g(f(x))$:
Leads to $g(a) \approx g(f(a)) \approx g(f(f(a))) \approx \cdots$

Direction of the identities in E may affect termination: e.g. $g(f(x)) \approx g(x)$.

No good theory exists for understanding termination.

Special Case: Grounded Identities

Theorem if all identities in E are grounded, then equality saturation terminates, and the ground word problem is decidable.

Proof: start with the E-graph consisting of s, t, and all terms in E.

Then we never need to insert a new nodes in G, hence the congruence closure algorithm terminates.

Example from [Baader and Nipkow, 1999]

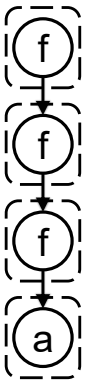
$$E = \boxed{\begin{array}{l} f(f(a)) \approx a \\ f(f(f(a))) \approx a \end{array}}$$

Check if $\boxed{f(a) \approx_E a}$

Example from [Baader and Nipkow, 1999]

$E = \boxed{\begin{array}{l} f(f(a)) \approx a \\ f(f(f(a))) \approx a \end{array}}$

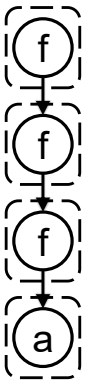
Check if $\boxed{f(a) \approx_E a}$



Example from [Baader and Nipkow, 1999]

$E = \boxed{\begin{array}{l} f(f(a)) \approx a \\ f(f(f(a))) \approx a \end{array}}$

Check if $\boxed{f(a) \approx_E a}$

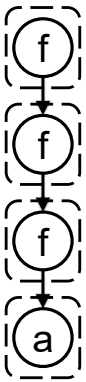


$f(f(f(a))) \approx a$

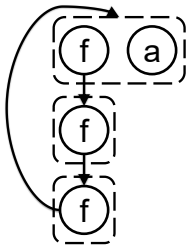
Example from [Baader and Nipkow, 1999]

$E = \boxed{\begin{array}{l} f(f(a)) \approx a \\ f(f(f(a))) \approx a \end{array}}$

Check if $\boxed{f(a) \approx_E a}$



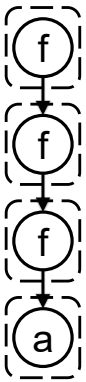
$f(f(f(a))) \approx a$



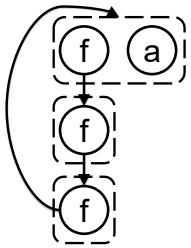
Example from [Baader and Nipkow, 1999]

$E = \boxed{\begin{array}{l} f(f(a)) \approx a \\ f(f(f(a))) \approx a \end{array}}$

Check if $\boxed{f(a) \approx_E a}$



$f(f(f(a))) \approx a$

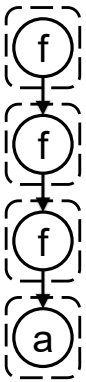


$f(f(a)) \approx a$

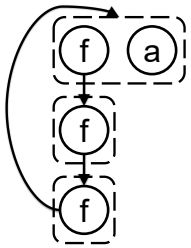
Example from [Baader and Nipkow, 1999]

$E = \boxed{\begin{array}{l} f(f(a)) \approx a \\ f(f(f(a))) \approx a \end{array}}$

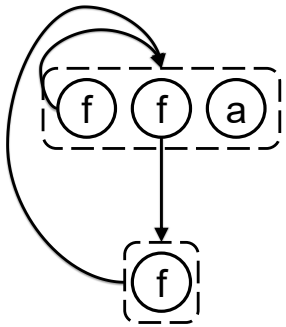
Check if $\boxed{f(a) \approx_E a}$



$f(f(f(a))) \approx a$



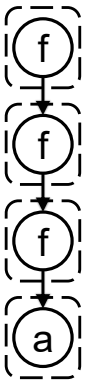
$f(f(a)) \approx a$



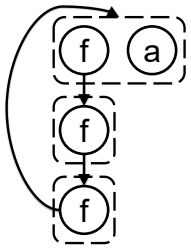
Example from [Baader and Nipkow, 1999]

$E = \boxed{\begin{array}{l} f(f(a)) \approx a \\ f(f(f(a))) \approx a \end{array}}$

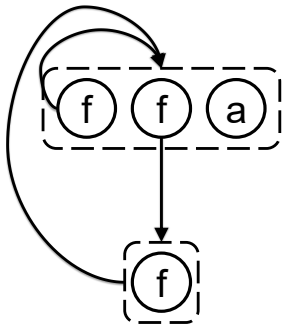
Check if $\boxed{f(a) \approx_E a}$



$f(f(f(a))) \approx a$



$f(f(a)) \approx a$

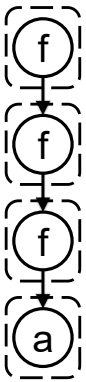


$f(f(f(a))) \approx a$

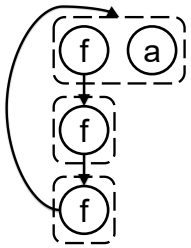
Example from [Baader and Nipkow, 1999]

$E = \boxed{\begin{array}{l} f(f(a)) \approx a \\ f(f(f(a))) \approx a \end{array}}$

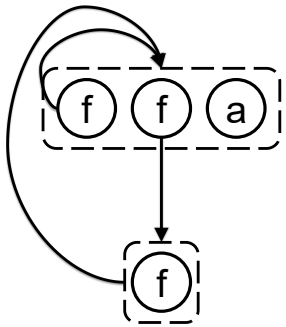
Check if $\boxed{f(a) \approx_E a}$



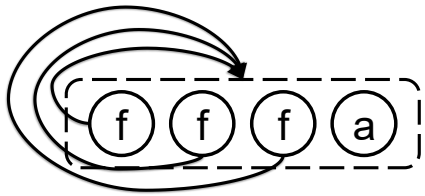
$f(f(f(a))) \approx a$



$f(f(a)) \approx a$



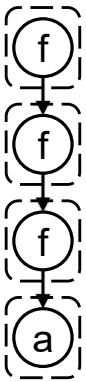
$f(f(f(a))) \approx a$



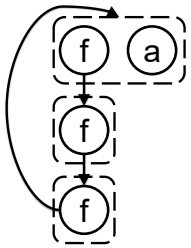
Example from [Baader and Nipkow, 1999]

$E = \boxed{\begin{array}{l} f(f(a)) \approx a \\ f(f(f(a))) \approx a \end{array}}$

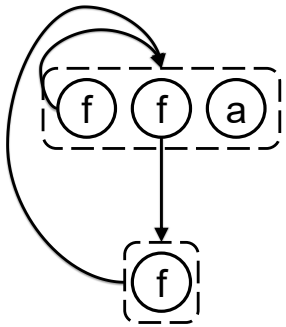
Check if $\boxed{f(a) \approx_E a}$



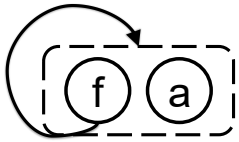
$f(f(f(a))) \approx a$



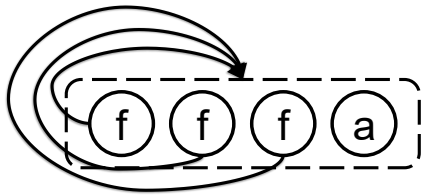
$f(f(a)) \approx a$



$f(f(f(a))) \approx a$



$\Leftarrow$



Discussion

The extensible query optimizers Volcano and Cascades are essentially E-graphs on steroids.

However, they come with their specialized terminology, and with heavy engineering.

Will discuss them in the next lecture(s), and refer to TRS and E-graphs.


Baader, F. and Nipkow, T. (1999).
Term Rewriting and All That.
Cambridge University Press, USA.


Willsey, M., Nandi, C., Wang, Y. R., Flatt, O., Tatlock, Z., and Panchekha, P. (2021).
egg: Fast and extensible equality saturation.
Proc. ACM Program. Lang., 5(POPL):1–29.