Rewriting

Part 6. Completion of Term Rewriting Systems

Temur Kutsia

RISC, JKU Linz

Word problem

Recall the word problem:

Given: A set of identities E and two terms s and t.

Decide: $s \approx_E t$ holds or not.

Word problem

Recall the word problem:

Given: A set of identities E and two terms s and t.

Decide: $s \approx_E t$ holds or not.

- The problem is undecidable for an arbitrary E.

Word problem

Recall the word problem:

Given: A set of identities E and two terms s and t.

Decide: $s \approx_E t$ holds or not.

- The problem is undecidable for an arbitrary E.
- Try to construct a decision procedure for a given finite E.

Word problem

Recall the word problem:

Given: A set of identities E and two terms s and t.

Decide: $s \approx_E t$ holds or not.

- The problem is undecidable for an arbitrary E.
- Try to construct a decision procedure for a given finite E.
- When E is finite and $\to_E$ is convergent, the word problem is decidable.

First Approach

Construction of a decision procedure.

First Approach

Construction of a decision procedure.

Show Termination: Try to find a reduction order $>$ which orients all identities in E. If this succeeds, consider the TRS $R := \{s \to t \mid s \approx t \in E \text{ or } t \approx s \in E, \text{ and } s > t\}$, and continue with this system in the next step. Otherwise fail.

First Approach

Construction of a decision procedure.

Show Termination: Try to find a reduction order $>$ which orients all identities in E. If this succeeds, consider the TRS $R := \{s \to t \mid s \approx t \in E$ or $t \approx s \in E$, and $s > t\}$, and continue with this system in the next step. Otherwise fail.

Show Confluence: Decide confluence of the terminating TRS R, by computing all critical pairs between rules in R and testing them for confluence. If this step succeeds, the rewrite relation $\to_R$ yields a decision procedure for the word problem for E. Otherwise fail.

Example When The Simple Approach Succeeds

Example 6.1

Let $E := \{x + 0 \approx x,\ \ x + s(y) \approx s(x + y)\}$.

Example When The Simple Approach Succeeds

Example 6.1

Let $E := \{x + 0 \approx x,\ x + s(y) \approx s(x + y)\}$.

Show Termination: Use the lpo $>_{lpo}$ induced by $+ > s$. We get a terminating term rewriting system

$$R := \{x + 0 \to x, x + s(y) \to s(x + y)\}.$$

Example When The Simple Approach Succeeds

Example 6.1

Let $E := \{x + 0 \approx x,\ x + s(y) \approx s(x + y)\}$.

Show Termination: Use the lpo $>_{lpo}$ induced by $+ > s$. We get a terminating term rewriting system

$$R := \{x + 0 \to x, x + s(y) \to s(x + y)\}.$$

Show Confluence: It is also confluent since there are no critical pairs.

Example When The Simple Approach Does Not Succeed

Example 6.2

Let $E := \{x + 0 \approx x,\ x + s(y) \approx s(x + y)\}$.

Example When The Simple Approach Does Not Succeed

Example 6.2

Let $E := \{x + 0 \approx x,\ x + s(y) \approx s(x + y)\}$.

Show Termination: Now we use the lpo $>_{lpo}$ induced by $s > +$. We get a terminating term rewriting system

$$R := \{x + 0 \to x, s(x + y) \to x + s(y)\}.$$

Example When The Simple Approach Does Not Succeed

Example 6.2

Let $E := \{x + 0 \approx x,\ x + s(y) \approx s(x + y)\}$.

Show Termination: Now we use the lpo $>_{lpo}$ induced by $s > +$. We get a terminating term rewriting system

$$R := \{x + 0 \to x, s(x + y) \to x + s(y)\}.$$

Show Confluence: It is not confluent since the following critical pair is not joinable:

$$s(x + 0)$$

$$x + s(0) \qquad\qquad s(x)$$

Main Ideas Behind Completion

- If the critical pair $\langle s, t\rangle$ of R is not joinable, then there are distinct normal forms $\hat{s}, \hat{t}$ of s, t.
- Adding $\hat{s} \to \hat{t}$ or $\hat{t} \to \hat{s}$ does not change the equational theory generated by R, because $\hat{s} \approx \hat{t}$ is an equational consequence of R.
- In the extended system, $\langle s, t\rangle$ is joinable.
- To obtain a terminating new system, we need $\hat{s} > \hat{t}$ or $\hat{t} > \hat{s}$

The Basic Completion Procedure

Input:
A finite set E of Σ-identities and a reduction order $>$ on $T(\Sigma, V)$.

Output:
A finite convergent TRS R that is equivalent to E, if the procedure terminates successfully;
"`Fail`", if the procedure terminates unsuccessfully.

Initialization:
If there exists $(s \approx t) \in E$ such that $s \neq t$, $s \not> t$ and $t \not> s$, then terminate with output `Fail`.
Otherwise, $i := 0$ and $R_0 := \{l \to r \mid (l \approx r) \in E \cup E^{-1} \wedge l > r\}$.

`repeat` $R_{i+1} := R_i$;

  `for all` $\langle s, t \rangle \in CP(R_i)$ `do`

  (a) Reduce s, t to some R_i-normal forms $\widehat{s}, \widehat{t}$;
  (b) If $\widehat{s} \neq \widehat{t}$ and neither $\widehat{s} > \widehat{t}$ nor $\widehat{t} > \widehat{s}$, then terminate with output `Fail`;
  (c) If $\widehat{s} > \widehat{t}$, then $R_{i+1} := R_{i+1} \cup \{\widehat{s} \to \widehat{t}\}$;
  (d) If $\widehat{t} > \widehat{s}$, then $R_{i+1} := R_{i+1} \cup \{\widehat{t} \to \widehat{s}\}$;
  `od`

  $i := i + 1$;
`until` $R_i = R_{i-1}$;
`output` R_i;

The Basic Completion Procedure

The procedure shows three different types of behavior, depending on particular input E and $>$:

1. It may terminate with failure because one of the nontrivial input identities can not be ordered using $>$, or the normal forms of the terms in one of the critical pairs are distinct and can not be oriented by using $>$. Not much is gained. One can restart the procedure with a different reduction order.

The Basic Completion Procedure

The procedure shows three different types of behavior, depending on particular input E and $>$:

1. It may terminate with failure because one of the nontrivial input identities can not be ordered using $>$, or the normal forms of the terms in one of the critical pairs are distinct and can not be oriented by using $>$. Not much is gained. One can restart the procedure with a different reduction order.
2. It may terminate successfully with output R_n because in nth step of the iteration all critical pairs are joinable. R_n is a finite convergent system equivalent to E. It can be used to decide the word problem for E.

The Basic Completion Procedure

The procedure shows three different types of behavior, depending on particular input E and $>$:

1. It may terminate with failure because one of the nontrivial input identities can not be ordered using $>$, or the normal forms of the terms in one of the critical pairs are distinct and can not be oriented by using $>$. Not much is gained. One can restart the procedure with a different reduction order.
2. It may terminate successfully with output R_n because in nth step of the iteration all critical pairs are joinable. R_n is a finite convergent system equivalent to E. It can be used to decide the word problem for E.
3. It may run forever since infinitely many new rules are generated. In this case, $R_\infty := \bigcup_{i \geq 0} R_i$ is an infinite convergent system that is equivalent to E. Yields a semidecision procedure for $\approx_E$.

Example: The Procedure Terminates Successfully

Input:

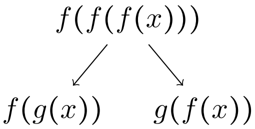
$E := \{f(f(x)) \approx g(x)\}$, LPO $>_{lpo}$ induced by $f > g$.

Example: The Procedure Terminates Successfully

Input:

$E := \{f(f(x)) \approx g(x)\}$, LPO $>_{lpo}$ induced by $f > g$.

$R_0 := \{f(f(x)) \rightarrow g(x)\}$ has a non-joinable critical pair:



Example: The Procedure Terminates Successfully

Input:

$E := \{f(f(x)) \approx g(x)\}$, LPO $>_{lpo}$ induced by $f > g$.

$R_0 := \{f(f(x)) \to g(x)\}$ has a non-joinable critical pair:

$$f(g(x)) \leftarrow f(f(f(x))) \rightarrow g(f(x))$$

$R_1 := \{f(f(x)) \to g(x), f(g(x)) \to g(f(x))\}$ is confluent.

$R_2 = R_1$.

Example: The Procedure Terminates Successfully

Input:

$E := \{f(f(x)) \approx g(x)\}$, LPO $>_{lpo}$ induced by $f > g$.

$R_0 := \{f(f(x)) \to g(x)\}$ has a non-joinable critical pair:

$$f(f(f(x)))$$

$$\swarrow \qquad \searrow$$

$$f(g(x)) \qquad g(f(x))$$

$R_1 := \{f(f(x)) \to g(x), f(g(x)) \to g(f(x))\}$ is confluent.

$R_2 = R_1$.

Output:

$R_2 := \{f(f(x)) \to g(x), f(g(x)) \to g(f(x))\}$.

Example: The Procedure Terminates with Failure

Input:

$E := \{x * (y + z) \approx (x * y) + (x * z), (u + v) * w \approx (u * w) + (v * w)\}$,

LPO $>_{lpo}$ induced by $* > +$.

Example: The Procedure Terminates with Failure

Input:

$E := \{x * (y + z) \approx (x * y) + (x * z), (u + v) * w \approx (u * w) + (v * w)\}$,

LPO $>_{lpo}$ induced by $* > +$.

$R_0 := \{x * (y + z) \to (x * y) + (x * z), (u + v) * w \to (u * w) + (v * w)\}$
has a non-joinable critical pair:

$$(u + v) * (y + z)$$

$$\swarrow \qquad \searrow$$

$$((u + v) * y) + ((u + v) * z) \qquad (u * (y + z)) + (v * (y + z))$$

$$\downarrow * \qquad \downarrow *$$

$$((u * y)+(v * y))+((u * z)+(v * z)) \quad \not\succ \; \neq \; \not\prec \quad ((u * y)+(u * z))+((v * y)+(v * z))$$

Example: The Procedure Terminates with Failure

Input:

$E := \{x * (y + z) \approx (x * y) + (x * z), (u + v) * w \approx (u * w) + (v * w)\}$,
LPO $>_{lpo}$ induced by $* > +$.

$R_0 := \{x * (y + z) \to (x * y) + (x * z), (u + v) * w \to (u * w) + (v * w)\}$
has a non-joinable critical pair:

$$(u + v) * (y + z)$$

$$((u + v) * y) + ((u + v) * z) \qquad\qquad (u * (y + z)) + (v * (y + z))$$

$$\downarrow * \qquad\qquad\qquad\qquad \downarrow *$$

$$((u * y) + (v * y)) + ((u * z) + (v * z)) \;\; \not> \;\; \neq \;\; \not< \;\; ((u * y) + (u * z)) + ((v * y) + (v * z))$$

The procedure fails.

Example: The Procedure Does Not Terminate

Input:

$E := \{x + 0 \approx x,\ x + s(y) \approx s(x + y)\}$, LPO $>_{lpo}$ induced by $s > +$.

Example: The Procedure Does Not Terminate

Input:

$E := \{x + 0 \approx x,\ x + s(y) \approx s(x + y)\}$, LPO $>_{lpo}$ induced by $s > +$.

$R_0 := \{x + 0 \to x,\ x + s(y) \to s(x + y)\}$.

$R_1 := R_0 \cup \{x + s(0) \to s(x)\}$.

Example: The Procedure Does Not Terminate

Input:

$E := \{x + 0 \approx x,\ x + s(y) \approx s(x + y)\}$, LPO $>_{lpo}$ induced by $s > +$.

$R_0 := \{x + 0 \to x,\ x + s(y) \to s(x + y)\}$.

$R_1 := R_0 \cup \{x + s(0) \to s(x)\}$.

R_1 is not confluent since the following critical pair is not joinable:

$$x + s(s(0)) \leftarrow s(x + s(0)) \rightarrow s(s(x))$$

Example: The Procedure Does Not Terminate

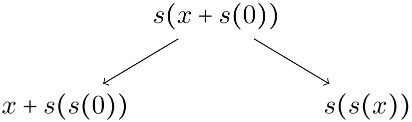
Input:

$E := \{x + 0 \approx x,\ x + s(y) \approx s(x + y)\}$, LPO $>_{lpo}$ induced by $s > +$.

$R_0 := \{x + 0 \to x,\ x + s(y) \to s(x + y)\}$.

$R_1 := R_0 \cup \{x + s(0) \to s(x)\}$.

R_1 is not confluent since the following critical pair is not joinable:



At each step of the iteration a new rule of the form $x + s^n(0) \to s^n(0)$ is generated. The procedure does not stop.

Drawbacks of the Basic Completion

- In practice, the basic completion procedure generates a huge number of rules.
- All of them should be taken into account when computing critical pairs.
- It makes both time and space requirement often unacceptably high.

Addressing the Drawbacks

- All implementations of completion “simplify” rules by reducing them with the help of other rules.
- If both sides of a rule reduce to the same term, the rule can be removed.
- Yields smaller rules.
- Improved completion procedure.

Example 6.3

$$R := \{f(f(x,y),z) \to f(x,f(y,z)),\ f(x,f(y,z)) \to f(x,z)\}$$

Addressing the Drawbacks

- All implementations of completion “simplify” rules by reducing them with the help of other rules.
- If both sides of a rule reduce to the same term, the rule can be removed.
- Yields smaller rules.
- Improved completion procedure.

Example 6.3

$R := \{f(f(x,y),z) \to f(x,f(y,z)),\ f(x,f(y,z)) \to f(x,z)\}$

$$
\begin{array}{ccc}
f(f(x,y),z) & \longrightarrow & f(x,f(y,z)) \\
 & & \downarrow \\
 & & f(x,z)
\end{array}
$$

Addressing the Drawbacks

- All implementations of completion "simplify" rules by reducing them with the help of other rules.
- If both sides of a rule reduce to the same term, the rule can be removed.
- Yields smaller rules.
- Improved completion procedure.

Example 6.3

$R := \{f(f(x,y),z) \to f(x,f(y,z)),\ f(x,f(y,z)) \to f(x,z)\}$

$$\begin{array}{ccc} f(f(x,y),z) & \longrightarrow & f(x,f(y,z)) \\ & & \downarrow \\ & & f(x,z) \end{array}$$

Simpler rules:

$R = \{f(f(x,y),z) \to f(x,z),\ f(x,f(y,z)) \to f(x,z)\}.$

An Improved Completion Procedure

- Described as a set of inference rules.
- Specific completion procedure is obtained by fixing a strategy for application of the rules.
- Works on pairs (E, R), where E is a set of identities and R is a set of rewrite rules.
- E contains input identities and not-yet-oriented critical pairs with the input reduction ordering $>$.
- R is a set of rewrite rules oriented with input ordering $>$.
- Goal: To transform an initial pair $(E_0, \varnothing)$ into $(\varnothing, R)$ such that R is convergent and equivalent to E.

An Improved Completion Procedure

DEDUCE	$\dfrac{E, R}{E \cup \{s \approx t\}, R}$	if $s \leftarrow_R u \rightarrow_R t$
ORIENT	$\dfrac{E \cup \{s \dot{\approx} t\}, R}{E, R \cup \{s \rightarrow t\}}$	if $s > t$
DELETE	$\dfrac{E \cup \{s \approx s\}, R}{E, R}$	
SIMPLIFY-IDENTITY	$\dfrac{E \cup \{s \dot{\approx} t\}, R}{E \cup \{u \approx t\}, R}$	if $s \rightarrow_R u$
R-SIMPLIFY-RULE	$\dfrac{E, R \cup \{s \rightarrow t\}}{E, R \cup \{s \rightarrow u\}}$	if $t \rightarrow_R u$
L-SIMPLIFY-RULE	$\dfrac{E, R \cup \{s \rightarrow t\}}{E \cup \{u \approx t\}, R}$	if $s \overset{\sqsupset}{\rightarrow}_R u$

An Improved Completion Procedure

- In the L-SIMPLIFY-RULE, $s \overset{\sqsupset}{\rightarrow}_R u$ says that s is reduced by a rule $l \rightarrow r \in R$ such that l can not be reduced by $s \rightarrow t$.

An Improved Completion Procedure

- In the L-SIMPLIFY-RULE, $s \overset{\sqsupset}{\rightarrow}_R u$ says that s is reduced by a rule $l \rightarrow r \in R$ such that l can not be reduced by $s \rightarrow t$.
- If $R := \{f(x,x) \rightarrow x, f(x,y) \rightarrow x\}$, then L-SIMPLIFY-RULE can be applied to $f(x,x) \rightarrow x$.

An Improved Completion Procedure

- In the L-SIMPLIFY-RULE, $s \xrightarrow{\sqsupset}_R u$ says that s is reduced by a rule $l \to r \in R$ such that l can not be reduced by $s \to t$.
- If $R := \{f(x,x) \to x, f(x,y) \to x\}$, then L-SIMPLIFY-RULE can be applied to $f(x,x) \to x$.
- If $R := \{f(x,y) \to x, f(x,y) \to y\}$, then L-SIMPLIFY-RULE can not be applied.

An Improved Completion Procedure

- In the L-SIMPLIFY-RULE, $s \stackrel{\sqsupset}{\rightarrow}_R u$ says that s is reduced by a rule $l \rightarrow r \in R$ such that l can not be reduced by $s \rightarrow t$.
- If $R := \{f(x,x) \rightarrow x, f(x,y) \rightarrow x\}$, then L-SIMPLIFY-RULE can be applied to $f(x,x) \rightarrow x$.
- If $R := \{f(x,y) \rightarrow x, f(x,y) \rightarrow y\}$, then L-SIMPLIFY-RULE can not be applied.
- Notation: $(E,R) \vdash_{\mathcal{C}} (E',R')$ means that (E,R) can be transformed into (E',R') by one of the inference rules.

Termination

Lemma 6.1 (Termination)

If $R \subseteq >$ *and* $(E, R) \vdash_{\mathcal{C}} (E', R')$*, then* $R' \subseteq >$*.*

Proof.

All rules are oriented wrt the reduction order $>$. $\square$

Soundness

Lemma 6.2 (Soundness)

If $(E_1, R_2) \vdash_{\mathcal{C}} (E_2, R_2)$, *then* $\approx_{E_1 \cup R_1} = \approx_{E_2 \cup R_2}$.

Soundness

Lemma 6.2 (Soundness)

If $(E_1, R_2) \vdash_{\mathcal{C}} (E_2, R_2)$, *then* $\approx_{E_1 \cup R_1} = \approx_{E_2 \cup R_2}$.

Proof.

Trivial for the first three rules.

Soundness

Lemma 6.2 (Soundness)

If $(E_1, R_2) \vdash_{\mathcal{C}} (E_2, R_2)$, *then* $\approx_{E_1 \cup R_1} = \approx_{E_2 \cup R_2}$.

Proof.

Trivial for the first three rules.

For Simplify-Identity, $E_1 = E \cup \{s \approx t\}$, $E_2 = E \cup \{u \approx t\}$, $R_1 = R = R_2$, and $s \to_R u$. We have $u \approx_{E_1 \cup R_1} t$, which implies $\approx_{E_2 \cup R_2} \subseteq \approx_{E_1 \cup R_1}$. Conversely, $u \approx t \in E_2$, $s \to_R u$, and $R = R_2$ imply that $s \approx_{E_2 \cup R_2} t$ and, hence, $\approx_{E_1 \cup R_1} \subseteq \approx_{E_2 \cup R_2}$.

□

Soundness

Lemma 6.2 (Soundness)

If $(E_1, R_2) \vdash_{\mathcal{C}} (E_2, R_2)$, *then* $\approx_{E_1 \cup R_1} = \approx_{E_2 \cup R_2}$.

Proof.

Trivial for the first three rules.

For SIMPLIFY-IDENTITY, $E_1 = E \cup \{s \approx t\}$, $E_2 = E \cup \{u \approx t\}$, $R_1 = R = R_2$, and $s \rightarrow_R u$. We have $u \approx_{E_1 \cup R_1} t$, which implies $\approx_{E_2 \cup R_2} \subseteq \approx_{E_1 \cup R_1}$. Conversely, $u \approx t \in E_2$, $s \rightarrow_R u$, and $R = R_2$ imply that $s \approx_{E_2 \cup R_2} t$ and, hence, $\approx_{E_1 \cup R_1} \subseteq \approx_{E_2 \cup R_2}$.

For R-SIMPLIFY, we have $E_1 = E = E_2$, $R_1 = R \cup \{s \to t\}$, $R_2 = R \cup \{s \to u\}$, and $t \rightarrow_R u$. $s \to t \in R_1$, $t \rightarrow_R u$, and $R \subseteq R_1$ imply $s \approx_{E_1 \cup R_1} u$. $s \to u \in R_2$, $t \rightarrow_R u$, and $R \subseteq R_2$ imply $s \approx_{E_2 \cup R_2} u$. Hence, $\approx_{E_1 \cup R_1} = \approx_{E_2 \cup R_2}$.

Soundness

Lemma 6.2 (Soundness)

If $(E_1, R_2) \vdash_{\mathcal{C}} (E_2, R_2)$*, then* $\approx_{E_1 \cup R_1} = \approx_{E_2 \cup R_2}$.

Proof.

Trivial for the first three rules.

For Simplify-Identity, $E_1 = E \cup \{s \approx t\}$, $E_2 = E \cup \{u \approx t\}$, $R_1 = R = R_2$, and $s \to_R u$. We have $u \approx_{E_1 \cup R_1} t$, which implies $\approx_{E_2 \cup R_2} \subseteq \approx_{E_1 \cup R_1}$. Conversely, $u \approx t \in E_2$, $s \to_R u$, and $R = R_2$ imply that $s \approx_{E_2 \cup R_2} t$ and, hence, $\approx_{E_1 \cup R_1} \subseteq \approx_{E_2 \cup R_2}$.

For R-Simplify, we have $E_1 = E = E_2$, $R_1 = R \cup \{s \to t\}$, $R_2 = R \cup \{s \to u\}$, and $t \to_R u$. $s \to t \in R_1$, $t \to_R u$, and $R \subseteq R_1$ imply $s \approx_{E_1 \cup R_1} u$. $s \to u \in R_2$, $t \to_R u$, and $R \subseteq R_2$ imply $s \approx_{E_2 \cup R_2} u$. Hence, $\approx_{E_1 \cup R_1} = \approx_{E_2 \cup R_2}$.

For L-Simplify the proof is similar. □

Completion Procedure

Definition 6.1 (Completion Procedure)

A completion procedure is a program that accepts as input a finite set of identities and a reduction order $>$, and uses the inference rules to generate a (finite or infinite) sequence

$$(E_0, R_0) \vdash_{\mathcal{C}} (E_1, R_1) \vdash_{\mathcal{C}} (E_2, R_2) \vdash_{\mathcal{C}} (E_3, R_3) \vdash_{\mathcal{C}} \cdots,$$

where $R_0 := 0$. The sequence is called a run of the procedure on input E_0 and $>$.

Completion Procedure

- To treat finite and infinite runs simultaneously, we extend every finite run $(E_0, R_0) \vdash_{\mathcal{C}} \cdots \vdash_{\mathcal{C}} (E_n, R_n)$ to an infinite one by setting $(E_{n+i}, R_{n+i}) := (E_n, R_n)$ for all $i \geq 1$.

Completion Procedure

- To treat finite and infinite runs simultaneously, we extend every finite run $(E_0, R_0) \vdash_{\mathcal{C}} \cdots \vdash_{\mathcal{C}} (E_n, R_n)$ to an infinite one by setting $(E_{n+i}, R_{n+i}) := (E_n, R_n)$ for all $i \geq 1$.
- Result of the run: persistent identities and rules:

$$E_\omega := \bigcup_{i \geq 0} \bigcap_{j \geq i} E_j \text{ and } R_\omega := \bigcup_{i \geq 0} \bigcap_{j \geq i} R_j.$$

Completion Procedure

- To treat finite and infinite runs simultaneously, we extend every finite run $(E_0, R_0) \vdash_{\mathcal{C}} \cdots \vdash_{\mathcal{C}} (E_n, R_n)$ to an infinite one by setting $(E_{n+i}, R_{n+i}) := (E_n, R_n)$ for all $i \geq 1$.
- Result of the run: persistent identities and rules:

$$E_\omega := \bigcup_{i \geq 0} \bigcap_{j \geq i} E_j \text{ and } R_\omega := \bigcup_{i \geq 0} \bigcap_{j \geq i} R_j.$$

- If the run is finite, then $E_\omega = E_n$ and $R_\omega = R_n$.

Completion Procedure

- To treat finite and infinite runs simultaneously, we extend every finite run $(E_0, R_0) \vdash_{\mathcal{C}} \cdots \vdash_{\mathcal{C}} (E_n, R_n)$ to an infinite one by setting $(E_{n+i}, R_{n+i}) := (E_n, R_n)$ for all $i \geq 1$.
- Result of the run: persistent identities and rules:

$$E_\omega := \bigcup_{i \geq 0} \bigcap_{j \geq i} E_j \text{ and } R_\omega := \bigcup_{i \geq 0} \bigcap_{j \geq i} R_j.$$

- If the run is finite, then $E_\omega = E_n$ and $R_\omega = R_n$.
- If the run is infinite, persistent identities (rules) are those that belong to some E_i (R_i) and are never removed in later inference steps.

Success, Failure, Correctness

Definition 6.2 (Success, Failure, Correctness)

A run on input E_o of a completion procedure

- succeeds iff $E_\omega = \emptyset$ and R_ω is convergent and equivalent to E_0,
- fails iff $E_\omega \neq \emptyset$,
- is correct iff every run that does not fail succeeds.

Success, Failure, Correctness

For the basic completion procedure,

- failure occurs if an input identity can not be oriented, or the normal forms of a critical pair are distinct (can not be removed by DELETE) and can not be oriented using $>$.
- The other two cases (terminates successfully, does not terminate) are successful in terms of Definition 6.2.

Success, Failure, Correctness

An arbitrary completion procedure may also have infinite failing runs.

Example 6.4

Input:

$$E_0 = \{h(x,y) \approx f(x), h(x,y) \approx f(y), f(g(f(x))) \approx f(g(x))\}$$

$>_{lpo}$ induced by $g > h > f > a$.

The procedure generates an infinite run with

$$E_\omega = \{f(x) \approx f(y)\}$$
$$R_\omega = \{h(x,y) \to f(x), h(x,y) \to f(y)\} \cup \{fg^n f(x) \to fg^n(x) \mid n \geq 1\}.$$

Success, Failure, Correctness

- It makes sense not to terminate with failure if a reduced and nonorientable identity is encountered.
- One simply defers the orientation of this identity until new rules are obtained.
- If the new set of rules allows one to simplify the identity to an orientable or trivial one, then one can apply ORIENT or DELETE.
- Otherwise, the treatment of this identity is deferred again.

Success, Failure, Correctness

Example 6.5

Input:

$E_0 = \{h(x,y) \approx f(x), h(x,y) \approx f(y), g(x,y) \approx h(x,y), g(x,y) \approx a\}$

$>_{lpo}$ induced by $g > h > f > a$.

Success, Failure, Correctness

Example 6.5

Input:

$E_0 = \{h(x,y) \approx f(x), h(x,y) \approx f(y), g(x,y) \approx h(x,y), g(x,y) \approx a\}$

$>_{lpo}$ induced by $g > h > f > a$.

Apply ORIENT 4 times:

$$E_4 = \emptyset$$
$$R_4 = \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y), g(x,y) \to a\}$$

Success, Failure, Correctness

Example 6.5

Input:

$E_0 = \{h(x,y) \approx f(x), h(x,y) \approx f(y), g(x,y) \approx h(x,y), g(x,y) \approx a\}$

$>_{lpo}$ induced by $g > h > f > a$.

Apply ORIENT 4 times:

$$E_4 = \emptyset$$
$$R_4 = \{h(x,y) \rightarrow f(x), h(x,y) \rightarrow f(y), \\ g(x,y) \rightarrow h(x,y), g(x,y) \rightarrow a\}$$

Apply DEDUCE twice:

$$E_6 = \{f(x) \approx f(y), h(x,y) \approx a\}$$
$$R_6 = \{h(x,y) \rightarrow f(x), h(x,y) \rightarrow f(y), \\ g(x,y) \rightarrow h(x,y), g(x,y) \rightarrow a\}$$

Success, Failure, Correctness

Example 6.6

Input:

$E_0 = \{h(x,y) \approx f(x), h(x,y) \approx f(y), g(x,y) \approx h(x,y), g(x,y) \approx a\}$

$>_{lpo}$ induced by $g > h > f > a$.

$$E_6 = \{f(x) \approx f(y), h(x,y) \approx a\}$$

$$R_6 = \{h(x,y) \to f(x), h(x,y) \to f(y),$$
$$g(x,y) \to h(x,y), g(x,y) \to a\}$$

Success, Failure, Correctness

Example 6.6

Input:

$E_0 = \{h(x,y) \approx f(x), h(x,y) \approx f(y), g(x,y) \approx h(x,y), g(x,y) \approx a\}$

$>_{lpo}$ induced by $g > h > f > a$.

$$E_6 = \{f(x) \approx f(y), h(x,y) \approx a\}$$
$$R_6 = \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y), g(x,y) \to a\}$$

Apply Orient:

$$E_7 = \{f(x) \approx f(y)\}$$
$$R_7 = \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y), g(x,y) \to a, h(x,y) \to a\}$$

Success, Failure, Correctness

Example 6.7

Input:

$E_0 = \{h(x,y) \approx f(x), h(x,y) \approx f(y), g(x,y) \approx h(x,y), g(x,y) \approx a\}$

$>_{lpo}$ induced by $g > h > f > a$.

$$E_7 = \{f(x) \approx f(y)\}$$

$$R_7 = \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y), g(x,y) \to a, h(x,y) \to a\}$$

Success, Failure, Correctness

Example 6.7

Input:

$E_0 = \{h(x,y) \approx f(x), h(x,y) \approx f(y), g(x,y) \approx h(x,y), g(x,y) \approx a\}$
$>_{lpo}$ induced by $g > h > f > a$.

$$E_7 = \{f(x) \approx f(y)\}$$
$$R_7 = \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y), g(x,y) \to a, h(x,y) \to a\}$$

Apply DEDUCE: (The basic completion would fail here, since the critical pair $f(x) \approx f(y)$ is unoriantable.)

$$E_8 = \{f(x) \approx f(y), f(x) \approx a\}$$
$$R_8 = \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y), g(x,y) \to a, h(x,y) \to a\}$$

Success, Failure, Correctness

Example 6.8

Input:

$E_0 = \{h(x,y) \approx f(x), h(x,y) \approx f(y), g(x,y) \approx h(x,y), g(x,y) \approx a\}$

$>_{lpo}$ induced by $g > h > f > a$.

$$E_8 = \{f(x) \approx f(y), f(x) \approx a\}$$

$$\begin{aligned} R_8 = \{&h(x,y) \to f(x), h(x,y) \to f(y), \\ &g(x,y) \to h(x,y), g(x,y) \to a, h(x,y) \to a\} \end{aligned}$$

Success, Failure, Correctness

Example 6.8

Input:

$E_0 = \{h(x,y) \approx f(x), h(x,y) \approx f(y), g(x,y) \approx h(x,y), g(x,y) \approx a\}$
$>_{lpo}$ induced by $g > h > f > a$.

$$E_8 = \{f(x) \approx f(y), f(x) \approx a\}$$
$$R_8 = \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y), g(x,y) \to a, h(x,y) \to a\}$$

Apply ORIENT

$$E_9 = \{f(x) \approx f(y)\}$$
$$R_9 = \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y) \; g(x,y) \to a, h(x,y) \to a, f(x) \to a\}$$

Success, Failure, Correctness

Example 6.9

Input:

$E_0 = \{h(x,y) \approx f(x), h(x,y) \approx f(y), g(x,y) \approx h(x,y), g(x,y) \approx a\}$

$>_{lpo}$ induced by $g > h > f > a$.

$$E_9 = \{f(x) \approx f(y)\}$$

$$R_9 = \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y)$$
$$g(x,y) \to a, h(x,y) \to a, f(x) \to a\}$$

Success, Failure, Correctness

Example 6.9

Input:

$E_0 = \{h(x,y) \approx f(x), h(x,y) \approx f(y), g(x,y) \approx h(x,y), g(x,y) \approx a\}$

$>_{lpo}$ induced by $g > h > f > a$.

$$E_9 = \{f(x) \approx f(y)\}$$
$$R_9 = \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y)$$
$$g(x,y) \to a, h(x,y) \to a, f(x) \to a\}$$

Apply Simplify-Identity twice

$$E_{11} = \{a \approx a\}$$
$$R_{11} = \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y)$$
$$g(x,y) \to a, h(x,y) \to a, f(x) \to a\}$$

Success, Failure, Correctness

Example 6.10

Input:

$E_0 = \{h(x,y) \approx f(x), h(x,y) \approx f(y), g(x,y) \approx h(x,y), g(x,y) \approx a\}$

$>_{lpo}$ induced by $g > h > f > a$.

$$E_{11} = \{a \approx a\}$$

$$R_{11} = \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y)$$
$$g(x,y) \to a, h(x,y) \to a, f(x) \to a\}$$

Success, Failure, Correctness

Example 6.10

Input:

$E_0 = \{h(x,y) \approx f(x), h(x,y) \approx f(y), g(x,y) \approx h(x,y), g(x,y) \approx a\}$
$>_{lpo}$ induced by $g > h > f > a$.

$$E_{11} = \{a \approx a\}$$
$$R_{11} = \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y)$$
$$g(x,y) \to a, h(x,y) \to a, f(x) \to a\}$$

Apply Delete

$$E_{11} = \emptyset$$
$$R_{11} = \{h(x,y) \to f(x), h(x,y) \to f(y), g(x,y) \to h(x,y)$$
$$g(x,y) \to a, h(x,y) \to a, f(x) \to a\}$$

Hence, we manage to simplify and delete an unorientable identity.

Fairness

Definition 6.3 (Fairness)

A run of a completion procedure is called fair iff

$$CP(R_\omega) \subseteq \bigcup_{i \geq 0} E_i.$$

A completion procedure is fair iff every non-failing run is fair.

Theorem 6.1

Every fair completion procedure is correct.