Completion Tools

Nao Hirokawa

JAIST

14th ISR, August 31, 2024

`https://www.jaist.ac.jp/~hirokawa/24isr/`

Contents

1 completion

2 automation

3 Horn clauses

Complete Presentations

Definition

- TRS is **complete** if it is terminating and confluent
- complete TRS $\mathcal{R}$ is **complete presentation** of ES $\mathcal{E}$ if $\leftrightarrow^*_{\mathcal{E}} = \leftrightarrow^*_{\mathcal{R}}$

Example

$$\left\{\begin{array}{c} \mathsf{s}(\mathsf{p}(x)) \approx x \\ \mathsf{p}(\mathsf{s}(x)) \approx x \\ \mathsf{s}(x) + y \approx \mathsf{s}(x+y) \end{array}\right\} \text{ admits complete presentation } \left\{\begin{array}{c} \mathsf{s}(\mathsf{p}(x)) \to x \\ \mathsf{p}(\mathsf{s}(x)) \to x \\ \mathsf{s}(x) + y \to \mathsf{s}(x+y) \\ \mathsf{p}(x) + y \to \mathsf{p}(x+y) \end{array}\right\}$$

how to find complete presentation? ☞ completion!

Knuth-Bendix Completion Procedure (1970)

input: equational system $\mathcal{E}$ and reduction order $>$
output: complete presentation $\mathcal{R}$ of $\mathcal{E}$

$\mathcal{R} := \varnothing$; $C := \mathcal{E}$;
while $C \neq \varnothing$ do
  choose $s \approx t \in C$;
  $C := C \setminus \{s \approx t\}$;
  normalize s and t to s' and t' with respect to $\mathcal{R}$;
  if $s' = t'$ then $\mathcal{S} = \varnothing$
  else if $s' > t'$ then $\mathcal{S} = \{s' \to t'\}$
  else if $t' > s'$ then $\mathcal{S} = \{t' \to s'\}$
  else *failure*
  fi;
  $C := C \cup \mathsf{CP}(\mathcal{R}, \mathcal{S}) \cup \mathsf{CP}(\mathcal{S}, \mathcal{R}) \cup \mathsf{CP}(\mathcal{S})$;
  $\mathcal{R} := \mathcal{R} \cup \mathcal{S}$
od

Abstract Completion

Definition (Bachmair et al. 1986)

given reduction order $>$

deduce	$\mathcal{E},\mathcal{R} \vdash \mathcal{E}\cup\{t\approx u\},\mathcal{R}$	if $t \;{}_{\mathcal{R}}\!\leftarrow \cdot \rightarrow_{\mathcal{R}} u$
delete	$\mathcal{E}\uplus\{s\approx s\},\mathcal{R} \vdash \mathcal{E}$	
orient	$\mathcal{E}\uplus\{s\approx t\},\mathcal{R} \vdash \mathcal{E},\mathcal{R}\cup\{s\to t\}$	if $s>t$
simplify	$\mathcal{E}\uplus\{s\approx t\},\mathcal{R} \vdash \mathcal{E}\cup\{s'\approx t\},\mathcal{R}$	if $s\to_{\mathcal{R}} s'$
collapse	$\mathcal{E},\mathcal{R}\uplus\{s\to t\} \vdash \mathcal{E}\cup\{s'\approx t\},\mathcal{R}$	if $s\to_{\mathcal{R}} s'$
compose	$\mathcal{E},\mathcal{R}\uplus\{s\to t\} \vdash \mathcal{E},\mathcal{R}\cup\{s\to t'\}$	if $t\to_{\mathcal{R}} t'$

Theorem

$\mathcal{R}_n$ is complete presentation of $\mathcal{E}_0$ if

$(\mathcal{E}_0,\mathcal{R}_0)\vdash\cdots\vdash(\mathcal{E}_n,\mathcal{R}_n)$ with $\mathcal{R}_0=\mathcal{E}_n=\varnothing$ and $\mathsf{CP}(\mathcal{R})\subseteq\mathcal{E}_0\cup\cdots\cup\mathcal{E}_n$

Example of Completion (LPO with $+ \succ \mathsf{s} \succ \mathsf{p}$)

$\mathsf{s}(x)+y\to\mathsf{s}(x+y)$

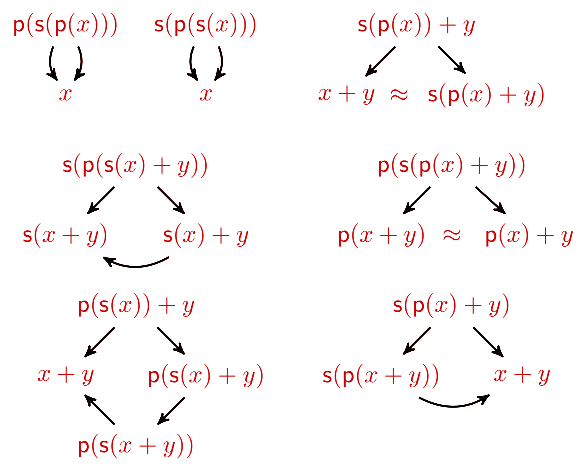
$\mathsf{s}(\mathsf{p}(x))\to x$

$\mathsf{p}(\mathsf{s}(x))\to x$

$x+y\approx x+y$

$\mathsf{p}(x+y)\leftarrow\mathsf{p}(x)+y$

canonical TRS



Canonical Presentations

Definition

- $\mathcal{R}$ is reduced if $\forall \ell\to r\in\mathcal{R}.\ \begin{cases}\ell\in\mathsf{NF}(\mathcal{R}\setminus\{\ell\to r\})\\ r\in\mathsf{NF}(\mathcal{R})\end{cases}$
- complete TRS is canonical if it is reduced

Example

following complete TRS is not reduced

$$\begin{array}{lll}\mathsf{s}(\mathsf{p}(x))\to x & \mathsf{s}(x)+y\to\mathsf{s}(x+y) & \mathsf{s}(\mathsf{p}(x)+y)\to x+y\\ \mathsf{p}(\mathsf{s}(x))\to x & \mathsf{p}(x)+y\to\mathsf{p}(x+y) & \end{array}$$

how to obtain canonical TRS? ☞ inter-reduction

Abstract Completion

Definition (Bachmair et al. 1986)

given reduction order $>$

deduce	$\mathcal{E},\mathcal{R} \vdash \mathcal{E}\cup\{t\approx u\},\mathcal{R}$	if $t \;{}_{\mathcal{R}}\!\leftarrow \cdot \rightarrow_{\mathcal{R}} u$
delete	$\mathcal{E}\uplus\{s\approx s\},\mathcal{R} \vdash \mathcal{E}$	
orient	$\mathcal{E}\uplus\{s\approx t\},\mathcal{R} \vdash \mathcal{E},\mathcal{R}\cup\{s\to t\}$	if $s>t$
simplify	$\mathcal{E}\uplus\{s\approx t\},\mathcal{R} \vdash \mathcal{E}\cup\{s'\approx t\},\mathcal{R}$	if $s\to_{\mathcal{R}} s'$
collapse	$\mathcal{E},\mathcal{R}\uplus\{s\to t\} \vdash \mathcal{E}\cup\{s'\approx t\},\mathcal{R}$	if $s\to_{\mathcal{R}} s'$
compose	$\mathcal{E},\mathcal{R}\uplus\{s\to t\} \vdash \mathcal{E},\mathcal{R}\cup\{s\to t'\}$	if $t\to_{\mathcal{R}} t'$

Theorem

$\mathcal{R}_n$ is complete presentation of $\mathcal{E}_0$ if

$(\mathcal{E}_0,\mathcal{R}_0)\vdash\cdots\vdash(\mathcal{E}_n,\mathcal{R}_n)$ with $\mathcal{R}_0=\mathcal{E}_n=\varnothing$ and $\mathsf{CP}(\mathcal{R})\subseteq\mathcal{E}_0\cup\cdots\cup\mathcal{E}_n$

Completion with Inter-Reduction

$$\mathsf{s}(x) + y \to \mathsf{s}(x + y)$$
$$\mathsf{s}(\mathsf{p}(x)) \to x$$
$$\mathsf{p}(\mathsf{s}(x)) \to x$$
$$x + y \approx x + y$$
$$\mathsf{p}(x) + y \to \mathsf{p}(x + y)$$

canonical TRS

Properties of Canonical TRSs

Definition

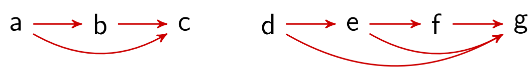
TRSs $\mathcal{R}$ and $\mathcal{S}$ are normalization equivalent if $\to^!_{\mathcal{R}}$ and $\to^!_{\mathcal{S}}$ coincide

Theorem (Ballantyne 1980, Métivier 1983, Hirokawa et al. 2019)

1. *every complete TRS admits normalization-equivalent canonical TRS*
2. *normalization-equivalent reduced TRSs are unique*
3. *among normalization-equivalent complete TRSs, canonical one is smallest*

Exercise: Completion as Union Find Algorithm

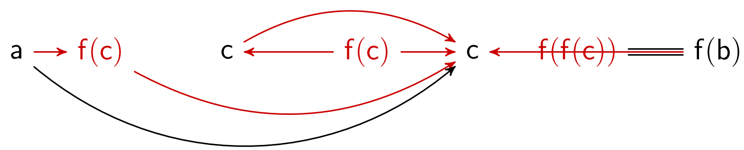
using $\mathsf{a} > \cdots > \mathsf{g}$



canonical presentation

Completion as Congruence Closure

using LPO with $\mathsf{a} > \mathsf{b} > \mathsf{c} > \mathsf{f}$



canonical TRS

Automation

How to Find Reduction Order?

$$\mathcal{E} = \left\{ \begin{array}{c} 0 + x \approx x \\ (-x) + x \approx 0 \\ (x + y) + z \approx x + (y + z) \end{array} \right\}$$

LPO with $- \succ + \succ 0$ → **COMPLETION**

$$\mathcal{R} = \left\{ \begin{array}{rr} 0 + x \to x & -(-x) \to x \\ x + 0 \to x & x + ((-x) + y) \to y \\ (-x) + x \to 0 & (-x) + (x + y) \to y \\ x + (-x) \to 0 & -(x + y) \to (-y) + (-x) \\ -0 \to 0 & (x + y) + z \to x + (y + z) \end{array} \right\}$$

Existing Approaches to Find Reduction Orders

1. incremental construction — fokelore
2. exhaustive search with efficient data structure — Kondo and Kurihara 2000
3. incremental construction of two reduction orders — Sternagel and Zankl 2012
4. SMT solving — Klein and Hirokawa 2011; Sato and Winkler 2015

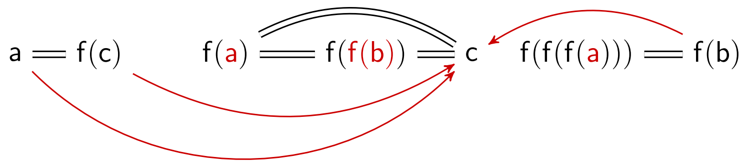
Exercise: Find Complete Presentation

construct complete presentation $\mathcal{R}$ by orienting some of equations $\mathcal{E}$:

$$\begin{array}{cc} \mathsf{s}(\mathsf{p}(x)) \to x & \mathsf{s}(x) + y \to \mathsf{s}(x + y) \\ \mathsf{p}(\mathsf{s}(x)) \to x & x + y \approx \mathsf{s}(\mathsf{p}(x) + y) \\ \mathsf{s}(\mathsf{p}(x) + y) \approx \mathsf{s}(x + y) & \mathsf{p}(x + y) \leftarrow \mathsf{p}(x) + y \\ \mathsf{p}((\mathsf{s}(x) + y) \approx \mathsf{s}(\mathsf{p}(x) + y) & \end{array}$$

in fact, $\mathcal{R}$ is oriented by LPO with $\mathsf{s} \succ \mathsf{p} \succ +$, and $s \downarrow_{\mathcal{R}} t$ for all $s \approx t \in \mathcal{E}$

Exercise: Find Canonical Presentation



how to automate this? ☞ MaxSAT/MaxSMT encoding

Problem

input: equational system $\mathcal{E}$ and class $\mathcal{RO}$ of reduction orders

output: complete presentation $\mathcal{R}$ of $\mathcal{E}$ and reduction order $>$ in $\mathcal{RO}$ with $\mathcal{R} \subseteq \mathcal{E} \cup \mathcal{E}^{-1}$ and $\mathcal{R} \subseteq >$

Heuristics (Sato and Winkler 2015)

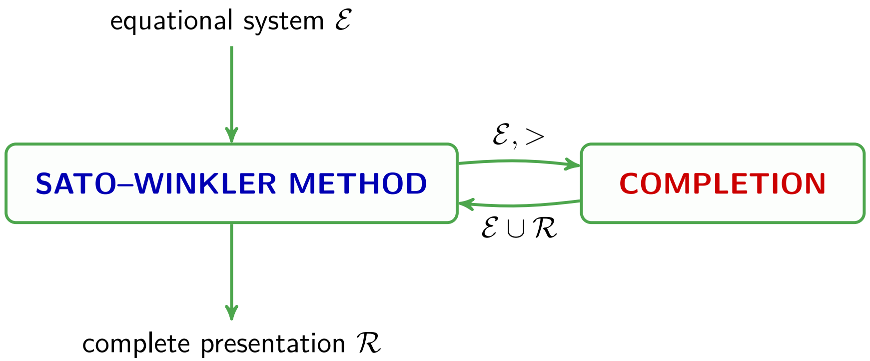
choose pair $(\mathcal{R}, >)$ that **minimizes** $|\mathcal{R}|$ subject to

$$s \notin \mathsf{NF}(\mathcal{R}) \text{ or } t \notin \mathsf{NF}(\mathcal{R}) \quad \text{for all } s \approx t \in \mathcal{E} \text{ with } s \neq t \qquad (*)$$

Rationale

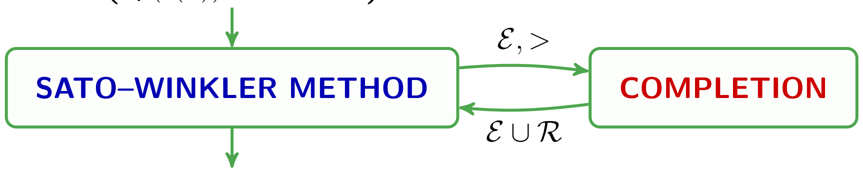
- $\mathcal{E} \subseteq \downarrow_{\mathcal{R}}$ implies $(*)$, and $(*)$ is SAT/SMT-encodable
- minimization aims to find **canonical** presentation

Flowchart of Completion



Example 1: Peano Arithmetic

$$\mathcal{E} = \left\{ \begin{array}{l} \mathsf{s}(x) + y \approx \mathsf{s}(x+y) \\ \mathsf{s}(\mathsf{p}(x)) \approx x \\ \mathsf{p}(\mathsf{s}(x)) \approx x \end{array} \right\}$$



$$\mathcal{R} = \left\{ \begin{array}{l} \mathsf{s}(x) + y \to \mathsf{s}(x+y) \\ \mathsf{s}(\mathsf{p}(x)) \to x \\ \mathsf{p}(\mathsf{s}(x)) \to x \\ \mathsf{p}(x) + y \to \mathsf{p}(x+y) \end{array} \right\}$$

```
plus(s(x),y) == s(plus(x,y))
s(p(x)) == x
p(s(x)) == x
```



```
plus(s(x),y) -> s(plus(x,y))
s(p(x)) -> x
p(s(x)) -> x
```

```
¦ LPO with precedence:
¦
¦ p > plus > s
```



```
plus(s(x),y) == s(plus(x,y))
s(p(x)) == x
p(s(x)) == x
plus(x0,x1) == s(plus(p(x0),x1))
```



```
plus(s(x),y) == s(plus(x,y))
s(p(x)) == x
p(s(x)) == x
plus(x0,x1) == s(plus(p(x0),x1))
```

```
plus(s(x),y) <- s(plus(x,y))
s(p(x)) -> x
p(s(x)) -> x
plus(x0,x1) == s(plus(p(x0),x1))
```

```
¦ LPO with precedence:
¦
¦ p > s > plus
```



```
plus(s(x),y) == s(plus(x,y))
s(p(x)) == x
p(s(x)) == x
plus(x0,x1) == s(plus(p(x0),x1))
p(plus(s(x0),x1)) == plus(x0,x1)
```



```
plus(s(x),y) == s(plus(x,y))
s(p(x)) == x
p(s(x)) == x
plus(x0,x1) == s(plus(p(x0),x1))
p(plus(s(x0),x1)) == plus(x0,x1)
```



```
plus(s(x),y) <- s(plus(x,y))
s(p(x)) -> x
p(s(x)) -> x
plus(x0,x1) == s(plus(p(x0),x1))
p(plus(s(x0),x1)) -> plus(x0,x1)
```

```
¦ LPO with precedence:
¦
¦ s > p > plus
```

```
plus(s(x),y) == s(plus(x,y))
s(p(x)) == x
p(s(x)) == x
plus(x0,x1) == s(plus(p(x0),x1))
p(plus(s(x0),x1)) == plus(x0,x1)
p(plus(x0,x1)) == plus(p(x0),x1)
p(plus(plus(s(x0),x1),x2)) == plus(plus(x0,x1),x2)
```



```
plus(s(x),y) == s(plus(x,y))
s(p(x)) == x
p(s(x)) == x
plus(x0,x1) == s(plus(p(x0),x1))
p(plus(s(x0),x1)) == plus(x0,x1)
p(plus(x0,x1)) == plus(p(x0),x1)
p(plus(plus(s(x0),x1),x2)) == plus(plus(x0,x1),x2)
```



```
plus(s(x),y) -> s(plus(x,y))
s(p(x)) -> x
p(s(x)) -> x
plus(x0,x1) == s(plus(p(x0),x1))
p(plus(s(x0),x1)) == plus(x0,x1)
p(plus(x0,x1)) <- plus(p(x0),x1)
p(plus(plus(s(x0),x1),x2)) == plus(plus(x0,x1),x2)
```

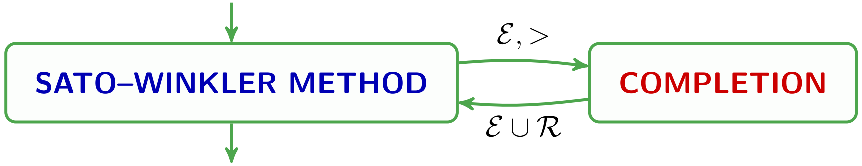
```
¦ LPO with precedence:
¦
¦ plus > s > p
```



```
YES

(VAR x0 x1 x y)
(RULES
  plus(p(x0),x1) -> p(plus(x0,x1))
  p(s(x)) -> x
  s(p(x)) -> x
  plus(s(x),y) -> s(plus(x,y))
)

(COMMENT
Termination is shown by LPO with precedence:

plus > s > p
)
```

Example 1: Peano Arithmetic

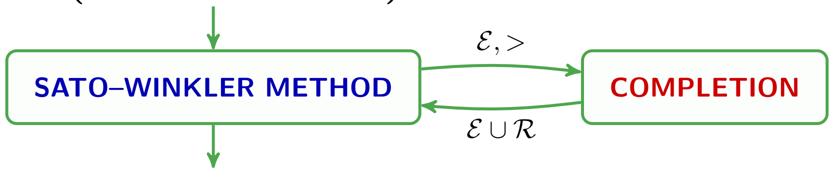
$$\mathcal{E} = \left\{ \begin{array}{l} \mathsf{s}(x) + y \approx \mathsf{s}(x+y) \\ \mathsf{s}(\mathsf{p}(x)) \approx x \\ \mathsf{p}(\mathsf{s}(x)) \approx x \end{array} \right\}$$



$$\mathcal{R} = \left\{ \begin{array}{l} \mathsf{s}(x) + y \to \mathsf{s}(x+y) \\ \mathsf{s}(\mathsf{p}(x)) \to x \\ \mathsf{p}(\mathsf{s}(x)) \to x \\ \mathsf{p}(x) + y \to \mathsf{p}(x+y) \end{array} \right\}$$

Example 2: Group

$$\mathcal{E} = \left\{ \begin{array}{c} 0 + x \approx x \\ (-x) + x \approx 0 \\ (x+y)+z \approx x + (y+z) \end{array} \right\}$$



$$\mathcal{R} = \left\{ \begin{array}{rr} 0 + x \to x & -(-x) \to x \\ x + 0 \to x & x + ((-x)+y) \to y \\ (-x) + x \to 0 & (-x) + (x+y) \to y \\ x + (-x) \to 0 & -(x+y) \to (-y) + (-x) \\ -0 \to 0 & (x+y)+z \to x + (y+z) \end{array} \right\}$$


```
+(0(),x) == x
+(-(x),x) == 0()
+(+(x,y),z) == +(x,+(y,z))
```



```
+(0(),x) -> x
+(-(x),x) -> 0()
+(+(x,y),z) -> +(x,+(y,z))
```

```
LPO with precedence:

- > + > 0
```

```
+(0(),x) == x
+(-(x),x) == 0()
+(+(x,y),z) == +(x,+(y,z))
x0 == +(-(x1),+(x1,x0))
```



```
+(0(),x) == x
+(-(x),x) == 0()
+(+(x,y),z) == +(x,+(y,z))
x0 == +(-(x1),+(x1,x0))
```



```
+(0(),x) -> x
+(-(x),x) -> 0()
+(+(x,y),z) -> +(x,+(y,z))
x0 <- +(-(x1),+(x1,x0))
```

```
¦ LPO with precedence:
¦
¦ - > + > 0
```



```
+(0(),x) == x
+(-(x),x) == 0()
+(+(x,y),z) == +(x,+(y,z))
x0 == +(-(x1),+(x1,x0))
+(-(0()),x0) == x0
+(-(-(x0)),0()) == x0
+(-(-(x0)),x1) == +(x0,x1)
+(-(+(x0,x1)),+(x0,+(x1,x2))) == x2
```

```
+(0(),x) == x
+(-(x),x) == 0()
+(+(x,y),z) == +(x,+(y,z))
x0 == +(-(x1),+(x1,x0))
+(-(0()),x0) == x0
+(-(-(x0)),0()) == x0
+(-(-(x0)),x1) == +(x0,x1)
+(-(+(x0,x1)),+(x0,+(x1,x2))) == x2
```



```
+(0(),x) -> x
+(-(x),x) -> 0()
+(+(x,y),z) -> +(x,+(y,z))
x0 <- +(-(x1),+(x1,x0))
+(-(0()),x0) -> x0
+(-(-(x0)),0()) == x0
+(-(-(x0)),x1) -> +(x0,x1)
+(-(+(x0,x1)),+(x0,+(x1,x2))) -> x2
```

```
¦ LPO with precedence:
¦
¦ - > + > 0
```



```
+(0(),x) == x
+(-(x),x) == 0()
+(+(x,y),z) == +(x,+(y,z))
x0 == +(-(x1),+(x1,x0))
+(-(0()),x0) == x0
+(-(-(x0)),0()) == x0
+(-(-(x0)),x1) == +(x0,x1)
+(-(+(x0,x1)),+(x0,+(x1,x2))) == x2
0() == -(0())
+(x0,0()) == x0
x0 == -(-(x0))
0() == +(x0,-(x0))
x0 == +(x1,+(-(x1),x0))
-(+(-(+(x0,x1)),x0)) == x1
+(-(+(x0,-(x1))),x0) == x1
+(-(+(x0,-(0()))),+(x0,x1)) == x1
+(-(+(-(+(x0,x1)),x0)),x2) == +(x1,x2)
+(-(+(x0,-(x1))),+(x0,x2)) == +(x1,x2)
+(-(+(x0,-(-(x1)))),+(x0,+(x1,x2))) == x2
+(-(+(x0,+(x1,x2))),+(x0,+(x1,+(x2,x3)))) == x3
+(-(+(x0,-(+(x1,x2)))),+(x0,x3)) == +(x1,+(x2,x3))
```



```
+(0(),x) == x
+(-(x),x) == 0()
+(+(x,y),z) == +(x,+(y,z))
x0 == +(-(x1),+(x1,x0))
+(-(0()),x0) == x0
+(-(-(x0)),0()) == x0
+(-(-(x0)),x1) == +(x0,x1)
+(-(+(x0,x1)),+(x0,+(x1,x2))) == x2
0() == -(0())
+(x0,0()) == x0
x0 == -(-(x0))
0() == +(x0,-(x0))
x0 == +(x1,+(-(x1),x0))
-(+(-(+(x0,x1)),x0)) == x1
+(-(+(x0,-(x1))),x0) == x1
+(-(+(x0,-(0()))),+(x0,x1)) == x1
+(-(+(-(+(x0,x1)),x0)),x2) == +(x1,x2)
+(-(+(x0,-(x1))),+(x0,x2)) == +(x1,x2)
+(-(+(x0,-(-(x1)))),+(x0,+(x1,x2))) == x2
+(-(+(x0,+(x1,x2))),+(x0,+(x1,+(x2,x3)))) == x3
+(-(+(x0,-(+(x1,x2)))),+(x0,x3)) == +(x1,+(x2,x3))
```

```
+(0(),x) -> x
+(-(x),x) -> 0()
+(+(x,y),z) -> +(x,+(y,z))
x0 <- +(-(x1),+(x1,x0))
+(-(0()),x0) == x0
+(-(-(x0)),0()) == x0
+(-(-(x0)),x1) == +(x0,x1)
+(-(+(x0,x1)),+(x0,+(x1,x2))) -> x2
0() <- -(0())
+(x0,0()) -> x0
x0 <- -(-(x0))
0() <- +(x0,-(x0))
x0 <- +(x1,+(-(x1),x0))
-(+(-(+(x0,x1)),x0)) -> x1
+(-(+(x0,-(x1))),x0) -> x1
+(-(+(x0,-(0()))),+(x0,x1)) == x1
+(-(+(-(+(x0,x1)),x0)),x2) == +(x1,x2)
+(-(+(x0,-(x1))),+(x0,x2)) -> +(x1,x2)
+(-(+(x0,-(-(x1)))),+(x0,+(x1,x2))) == x2
+(-(+(x0,+(x1,x2))),+(x0,+(x1,+(x2,x3)))) -> x3
+(-(+(x0,-(+(x1,x2)))),+(x0,x3)) == +(x1,+(x2,x3))
```

```
¦ LPO with precedence:
¦
¦ - > + > 0
```



```
+(0(),x) == x
+(-(x),x) == 0()
+(+(x,y),z) == +(x,+(y,z))
x0 == +(-(x1),+(x1,x0))
+(-(0()),x0) == x0
+(-(-(x0)),0()) == x0
+(-(-(x0)),x1) == +(x0,x1)
+(-(+(x0,x1)),+(x0,+(x1,x2))) == x2
0() == -(0())
+(x0,0()) == x0
x0 == -(-(x0))
0() == +(x0,-(x0))
x0 == +(x1,+(-(x1),x0))
-(+(-(+(x0,x1)),x0)) == x1
+(-(+(x0,-(x1))),x0) == x1
+(-(+(x0,-(0()))),+(x0,x1)) == x1
+(-(+(-(+(x0,x1)),x0)),x2) == +(x1,x2)
+(-(+(x0,-(x1))),+(x0,x2)) == +(x1,x2)
+(-(+(x0,-(-(x1)))),+(x0,+(x1,x2))) == x2
+(-(+(x0,+(x1,x2))),+(x0,+(x1,+(x2,x3)))) == x3
+(-(+(x0,-(+(x1,x2)))),+(x0,x3)) == +(x1,+(x2,x3))
+(x0,-(+(-(x1),x0))) == x1
+(-(+(x0,x1)),x0) == -(x1)
-(+(x0,-(+(x1,x0)))) == x1
-(+(x0,-(x1))) == +(x1,-(x0))
-(+(-(x0),-(x1))) == +(x1,x0)
-(+(-(x0),x1)) == +(-(x1),x0)
+(x0,+(x1,-(+(x0,x1)))) == 0()
+(x0,+(-(+(x1,x0)),x1)) == 0()
+(-(x0),-(+(-(x1),-(x0)))) == x1
-(+(-(x0),-(+(x1,-(x0))))) == x1
```

```
+(x0,+(x1,+(-(+(x0,x1)),x2))) == x2
+(x0,+(-(+(x1,x0)),+(x1,x2))) == x2
-(+(-(+(x0,+(x1,x2))),+(x0,x1))) == x2
+(-(+(x0,+(-(+(x1,x2)),x1))),x0) == x2
+(-(+(x0,+(x1,x2))),+(x0,x1)) == -(x2)
+(-(+(x0,+(x1,-(x2)))),+(x0,x1)) == x2
+(x0,+(-(+(-(x1),x0)),x2)) == +(x1,x2)
+(-(+(x0,x1)),+(x0,x2)) == +(-(x1),x2)
+(-(+(x0,x1)),x0) == +(-(+(x2,x1)),x2)
+(-(+(x0,-(x1))),x2) == +(x1,+(-(x0),x2))
+(-(+(-(x0),-(x1))),x2) == +(x1,+(x0,x2))
```



```
+(0(),x) == x
+(-(x),x) == 0()
+(+(x,y),z) == +(x,+(y,z))
x0 == +(-(x1),+(x1,x0))
+(-(0()),x0) == x0
+(-(-(x0)),0()) == x0
+(-(-(x0)),x1) == +(x0,x1)
+(-(+(x0,x1)),+(x0,+(x1,x2))) == x2
0() == -(0())
+(x0,0()) == x0
x0 == -(-(x0))
0() == +(x0,-(x0))
x0 == +(x1,+(-(x1),x0))
-(+(-(+(x0,x1)),x0)) == x1
+(-(+(x0,-(x1))),x0) == x1
+(-(+(x0,-(0()))),+(x0,x1)) == x1
+(-(+(-(+(x0,x1)),x0)),x2) == +(x1,x2)
+(-(+(x0,-(x1))),+(x0,x2)) == +(x1,x2)
+(-(+(x0,-(-(x1)))),+(x0,+(x1,x2))) == x2
+(-(+(x0,+(x1,x2))),+(x0,+(x1,+(x2,x3)))) == x3
+(-(+(x0,-(+(x1,x2)))),+(x0,x3)) == +(x1,+(x2,x3))
+(x0,-(+(-(x1),x0))) == x1
+(-(+(x0,x1)),x0) == -(x1)
-(+(x0,-(+(x1,x0)))) == x1
-(+(x0,-(x1))) == +(x1,-(x0))
-(+(-(x0),-(x1))) == +(x1,x0)
-(+(-(x0),x1)) == +(-(x1),x0)
+(x0,+(x1,-(+(x0,x1)))) == 0()
+(x0,+(-(+(x1,x0)),x1)) == 0()
+(-(x0),-(+(-(x1),-(x0)))) == x1
-(+(-(x0),-(+(x1,-(x0))))) == x1
```

```
+(x0,+(x1,+(-(+(x0,x1)),x2))) == x2
+(x0,+(-(+(x1,x0)),+(x1,x2))) == x2
-(+(-(+(x0,+(x1,x2))),+(x0,x1))) == x2
+(-(+(x0,+(-(+(x1,x2)),x1))),x0) == x2
+(-(+(x0,+(x1,x2))),+(x0,x1)) == -(x2)
+(-(+(x0,+(x1,-(x2)))),+(x0,x1)) == x2
+(x0,+(-(+(-(x1),x0)),x2)) == +(x1,x2)
+(-(+(x0,x1)),+(x0,x2)) == +(-(x1),x2)
+(-(+(x0,x1)),x0) == +(-(+(x2,x1)),x2)
+(-(+(x0,-(x1))),x2) == +(x1,+(-(x0),x2))
+(-(+(-(x0),-(x1))),x2) == +(x1,+(x0,x2))
```



```
+(0(),x) -> x
+(-(x),x) -> 0()
+(+(x,y),z) -> +(x,+(y,z))
x0 <- +(-(x1),+(x1,x0))
+(-(0()),x0) == x0
+(-(-(x0)),0()) == x0
+(-(-(x0)),x1) == +(x0,x1)
+(-(+(x0,x1)),+(x0,+(x1,x2))) == x2
0() <- -(0())
+(x0,0()) -> x0
x0 <- -(-(x0))
0() <- +(x0,-(x0))
x0 <- +(x1,+(-(x1),x0))
-(+(-(+(x0,x1)),x0)) == x1
+(-(+(x0,-(x1))),x0) == x1
+(-(+(x0,-(0()))),+(x0,x1)) == x1
+(-(+(-(+(x0,x1)),x0)),x2) == +(x1,x2)
+(-(+(x0,-(x1))),+(x0,x2)) == +(x1,x2)
+(-(+(x0,-(-(x1)))),+(x0,+(x1,x2))) == x2
+(-(+(x0,+(x1,x2))),+(x0,+(x1,+(x2,x3)))) == x3
+(-(+(x0,-(+(x1,x2)))),+(x0,x3)) == +(x1,+(x2,x3))
+(x0,-(+(-(x1),x0))) == x1
+(-(+(x0,x1)),x0) -> -(x1)
-(+(x0,-(+(x1,x0)))) == x1
-(+(x0,-(x1))) -> +(x1,-(x0))
-(+(-(x0),-(x1))) == +(x1,x0)
-(+(-(x0),x1)) -> +(-(x1),x0)
+(x0,+(x1,-(+(x0,x1)))) -> 0()
+(x0,+(-(+(x1,x0)),x1)) == 0()
+(-(x0),-(+(-(x1),-(x0)))) == x1
-(+(-(x0),-(+(x1,-(x0))))) == x1
```

```
+(x0,+(x1,+(-(+(x0,x1)),x2))) -> x2
+(x0,+(-(+(x1,x0)),+(x1,x2))) == x2
-(+(-(+(x0,+(x1,x2))),+(x0,x1))) == x2
+(-(+(x0,+(-(+(x1,x2)),x1))),x0) == x2
+(-(+(x0,+(x1,x2))),+(x0,x1)) == -(x2)
+(-(+(x0,+(x1,-(x2)))),+(x0,x1)) == x2
+(x0,+(-(+(-(x1),x0)),x2)) == +(x1,x2)
+(-(+(x0,x1)),+(x0,x2)) -> +(-(x1),x2)
+(-(+(x0,x1)),x0) == +(-(+(x2,x1)),x2)
+(-(+(x0,-(x1))),x2) == +(x1,+(-(x0),x2))
+(-(+(-(x0),-(x1))),x2) == +(x1,+(x0,x2))
```

```
¦ LPO with precedence:
¦
¦ - > + > 0
```

```
+(0(),x) == x
+(-(x),x) == 0()
+(+(x,y),z) == +(x,+(y,z))
x0 == +(-(x1),+(x1,x0))
+(-(0()),x0) == x0
+(-(-(x0)),0()) == x0
+(-(-(x0)),x1) == +(x0,x1)
+(-(+(x0,x1)),+(x0,+(x1,x2))) == x2
0() == -(0())
+(x0,0()) == x0
x0 == -(-(x0))
0() == +(x0,-(x0))
x0 == +(x1,+(-(x1),x0))
-(+(-(+(x0,x1)),x0)) == x1
+(-(+(x0,-(x1))),x0) == x1
+(-(+(x0,-(0()))),+(x0,x1)) == x1
+(-(+(-(+(x0,x1)),x0)),x2) == +(x1,x2)
+(-(+(x0,-(x1))),+(x0,x2)) == +(x1,x2)
+(-(+(x0,-(-(x1)))),+(x0,+(x1,x2))) == x2
+(-(+(x0,+(x1,x2))),+(x0,+(x1,+(x2,x3)))) == x3
+(-(+(x0,-(+(x1,x2)))),+(x0,x3)) == +(x1,+(x2,x3))
+(x0,-(+(-(x1),x0))) == x1
+(-(+(x0,x1)),x0) == -(x1)
-(+(x0,-(+(x1,x0)))) == x1
-(+(x0,-(x1))) == +(x1,-(x0))
-(+(-(x0),-(x1))) == +(x1,x0)
-(+(-(x0),x1)) == +(-(x1),x0)
+(x0,+(x1,-(+(x0,x1)))) == 0()
+(x0,+(-(+(x1,x0)),x1)) == 0()
+(-(x0),-(+(-(x1),-(x0)))) == x1
-(+(-(x0),-(+(x1,-(x0))))) == x1
```

```
+(x0,+(x1,+(-(+(x0,x1)),x2))) == x2
+(x0,+(-(+(x1,x0)),+(x1,x2))) == x2
-(+(-(+(x0,+(x1,x2))),+(x0,x1))) == x2
+(-(+(x0,+(-(+(x1,x2)),x1))),x0) == x2
+(-(+(x0,+(x1,x2))),+(x0,x1)) == -(x2)
+(-(+(x0,+(x1,-(x2)))),+(x0,x1)) == x2
+(x0,+(-(+(-(x1),x0)),x2)) == +(x1,x2)
+(-(+(x0,x1)),+(x0,x2)) == +(-(x1),x2)
+(-(+(x0,x1)),x0) == +(-(+(x2,x1)),x2)
+(-(+(x0,-(x1))),x2) == +(x1,+(-(x0),x2))
+(-(+(-(x0),-(x1))),x2) == +(x1,+(x0,x2))
+(x0,-(+(x1,x0))) == -(x1)
-(+(x0,x1)) == +(-(x1),-(x0))
+(x0,+(-(+(x1,x0)),x2)) == +(-(x1),x2)
+(-(x0),+(-(x1),x2)) == +(-(+(x1,x0)),x2)
-(+(x0,x1)) == +(-(x1),+(x2,-(+(x0,x2))))
+(x0,+(x1,+(x2,-(+(x0,+(x1,x2)))))) == 0()
-(+(x0,+(x1,-(x2)))) == +(x2,-(+(x0,x1)))
+(-(x0),x1) == -(+(x2,+(-(+(x1,x2)),x0)))
-(+(x0,+(-(x1),x2))) == +(-(x2),+(x1,-(x0)))
-(+(x0,+(x1,-(x2)))) == +(x2,+(-(x1),-(x0)))
+(x0,+(x1,+(x2,+(-(+(x0,+(x1,x2))),x3)))) == x3
+(-(x1),+(x0,+(-(+(x2,x0)),x3))) == +(-(+(x2,x1)),x3)
+(-(x0),+(x1,x2)) == +(-(+(x3,+(-(+(x1,x3)),x0))),x2)
```



```
+(0(),x) == x
+(-(x),x) == 0()
+(+(x,y),z) == +(x,+(y,z))
x0 == +(-(x1),+(x1,x0))
+(-(0()),x0) == x0
+(-(-(x0)),0()) == x0
+(-(-(x0)),x1) == +(x0,x1)
+(-(+(x0,x1)),+(x0,+(x1,x2))) == x2
0() == -(0())
+(x0,0()) == x0
x0 == -(-(x0))
0() == +(x0,-(x0))
x0 == +(x1,+(-(x1),x0))
-(+(-(+(x0,x1)),x0)) == x1
+(-(+(x0,-(x1))),x0) == x1
+(-(+(x0,-(0()))),+(x0,x1)) == x1
+(-(+(-(+(x0,x1)),x0)),x2) == +(x1,x2)
+(-(+(x0,-(x1))),+(x0,x2)) == +(x1,x2)
+(-(+(x0,-(-(x1)))),+(x0,+(x1,x2))) == x2
+(-(+(x0,+(x1,x2))),+(x0,+(x1,+(x2,x3)))) == x3
+(-(+(x0,-(+(x1,x2)))),+(x0,x3)) == +(x1,+(x2,x3))
+(x0,-(+(-(x1),x0))) == x1
+(-(+(x0,x1)),x0) == -(x1)
-(+(x0,-(+(x1,x0)))) == x1
-(+(x0,-(x1))) == +(x1,-(x0))
-(+(-(x0),-(x1))) == +(x1,x0)
-(+(-(x0),x1)) == +(-(x1),x0)
+(x0,+(x1,-(+(x0,x1)))) == 0()
+(x0,+(-(+(x1,x0)),x1)) == 0()
+(-(x0),-(+(-(x1),-(x0)))) == x1
-(+(-(x0),-(+(x1,-(x0))))) == x1
```

```
+(x0,+(x1,+(-(+(x0,x1)),x2))) == x2
+(x0,+(-(+(x1,x0)),+(x1,x2))) == x2
-(+(-(+(x0,+(x1,x2))),+(x0,x1))) == x2
+(-(+(x0,+(-(+(x1,x2)),x1))),x0) == x2
+(-(+(x0,+(x1,x2))),+(x0,x1)) == -(x2)
+(-(+(x0,+(x1,-(x2)))),+(x0,x1)) == x2
+(x0,+(-(+(-(x1),x0)),x2)) == +(x1,x2)
+(-(+(x0,x1)),+(x0,x2)) == +(-(x1),x2)
+(-(+(x0,x1)),x0) == +(-(+(x2,x1)),x2)
+(-(+(x0,-(x1))),x2) == +(x1,+(-(x0),x2))
+(-(+(-(x0),-(x1))),x2) == +(x1,+(x0,x2))
+(x0,-(+(x1,x0))) == -(x1)
-(+(x0,x1)) == +(-(x1),-(x0))
+(x0,+(-(+(x1,x0)),x2)) == +(-(x1),x2)
+(-(x0),+(-(x1),x2)) == +(-(+(x1,x0)),x2)
-(+(x0,x1)) == +(-(x1),+(x2,-(+(x0,x2))))
+(x0,+(x1,+(x2,-(+(x0,+(x1,x2)))))) == 0()
-(+(x0,+(x1,-(x2)))) == +(x2,-(+(x0,x1)))
+(-(x0),x1) == -(+(x2,+(-(+(x1,x2)),x0)))
-(+(x0,+(-(x1),x2))) == +(-(x2),+(x1,-(x0)))
-(+(x0,+(x1,-(x2)))) == +(x2,+(-(x1),-(x0)))
+(x0,+(x1,+(x2,+(-(+(x0,+(x1,x2))),x3)))) == x3
+(-(x1),+(x0,+(-(+(x2,x0)),x3))) == +(-(+(x2,x1)),x3)
+(-(x0),+(x1,x2)) == +(-(+(x3,+(-(+(x1,x3)),x0))),x2)
```



```
+(0(),x) -> x
+(-(x),x) -> 0()
+(+(x,y),z) -> +(x,+(y,z))
x0 <- +(-(x1),+(x1,x0))
+(-(0()),x0) == x0
+(-(-(x0)),0()) == x0
+(-(-(x0)),x1) == +(x0,x1)
+(-(+(x0,x1)),+(x0,+(x1,x2))) == x2
0() <- -(0())
+(x0,0()) -> x0
x0 <- -(-(x0))
0() <- +(x0,-(x0))
x0 <- +(x1,+(-(x1),x0))
-(+(-(+(x0,x1)),x0)) == x1
+(-(+(x0,-(x1))),x0) == x1
+(-(+(x0,-(0()))),+(x0,x1)) == x1
+(-(+(-(+(x0,x1)),x0)),x2) == +(x1,x2)
+(-(+(x0,-(x1))),+(x0,x2)) == +(x1,x2)
+(-(+(x0,-(-(x1)))),+(x0,+(x1,x2))) == x2
+(-(+(x0,+(x1,x2))),+(x0,+(x1,+(x2,x3)))) == x3
+(-(+(x0,-(+(x1,x2)))),+(x0,x3)) == +(x1,+(x2,x3))
+(x0,-(+(-(x1),x0))) == x1
+(-(+(x0,x1)),x0) == -(x1)
-(+(x0,-(+(x1,x0)))) == x1
-(+(x0,-(x1))) == +(x1,-(x0))
-(+(-(x0),-(x1))) == +(x1,x0)
-(+(-(x0),x1)) == +(-(x1),x0)
+(x0,+(x1,-(+(x0,x1)))) == 0()
+(x0,+(-(+(x1,x0)),x1)) == 0()
+(-(x0),-(+(-(x1),-(x0)))) == x1
-(+(-(x0),-(+(x1,-(x0))))) == x1
```

```
+(x0,+(x1,+(-(+(x0,x1)),x2))) == x2
+(x0,+(-(+(x1,x0)),+(x1,x2))) == x2
-(+(-(+(x0,+(x1,x2))),+(x0,x1))) == x2
+(-(+(x0,+(-(+(x1,x2)),x1))),x0) == x2
+(-(+(x0,+(x1,x2))),+(x0,x1)) == -(x2)
+(-(+(x0,+(x1,-(x2)))),+(x0,x1)) == x2
+(x0,+(-(+(-(x1),x0)),x2)) == +(x1,x2)
+(-(+(x0,x1)),+(x0,x2)) == +(-(x1),x2)
+(-(+(x0,x1)),x0) == +(-(+(x2,x1)),x2)
+(-(+(x0,-(x1))),x2) == +(x1,+(-(x0),x2))
+(-(+(-(x0),-(x1))),x2) == +(x1,+(x0,x2))
+(x0,-(+(x1,x0))) == -(x1)
-(+(x0,x1)) -> +(-(x1),-(x0))
+(x0,+(-(+(x1,x0)),x2)) == +(-(x1),x2)
+(-(x0),+(-(x1),x2)) == +(-(+(x1,x0)),x2)
-(+(x0,x1)) == +(-(x1),+(x2,-(+(x0,x2))))
+(x0,+(x1,+(x2,-(+(x0,+(x1,x2)))))) == 0()
-(+(x0,+(x1,-(x2)))) == +(x2,-(+(x0,x1)))
+(-(x0),x1) == -(+(x2,+(-(+(x1,x2)),x0)))
-(+(x0,+(-(x1),x2))) == +(-(x2),+(x1,-(x0)))
-(+(x0,+(x1,-(x2)))) == +(x2,+(-(x1),-(x0)))
+(x0,+(x1,+(x2,+(-(+(x0,+(x1,x2))),x3)))) == x3
+(-(x1),+(x0,+(-(+(x2,x0)),x3))) == +(-(+(x2,x1)),x3)
+(-(x0),+(x1,x2)) == +(-(+(x3,+(-(+(x1,x3)),x0))),x2)
```

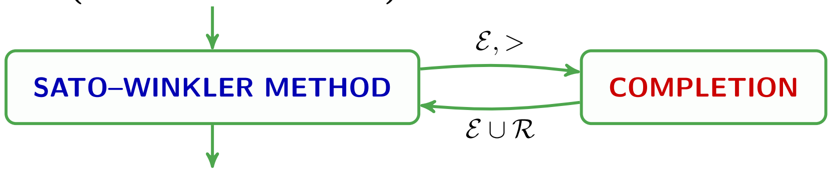
```
¦ LPO with precedence:
¦
¦ - > + > 0
```


■

```
YES

(VAR x0 x1 x y z)
(RULES
  -(+(x0,x1)) -> +(-(x1),-(x0))
  +(x1,+(-(x1),x0)) -> x0
  +(x0,-(x0)) -> 0()
  -(-(x0)) -> x0
  -(0()) -> 0()
  +(x0,0()) -> x0
  +(-(x1),+(x1,x0)) -> x0
  +(+(x,y),z) -> +(x,+(y,z))
  +(-(x),x) -> 0()
  +(0(),x) -> x
)

(COMMENT
Termination is shown by LPO with precedence:

- > + > 0
)
```

Example 2: Group

$$\mathcal{E} = \left\{ \begin{array}{c} 0 + x \approx x \\ (-x) + x \approx 0 \\ (x+y)+z \approx x+(y+z) \end{array} \right\}$$



$$\mathcal{R} = \left\{ \begin{array}{rr} 0 + x \to x & -(-x) \to x \\ x + 0 \to x & x + ((-x)+y) \to y \\ (-x)+x \to 0 & (-x)+(x+y) \to y \\ x+(-x) \to 0 & -(x+y) \to (-y)+(-x) \\ -0 \to 0 & (x+y)+z \to x+(y+z) \end{array} \right\}$$

Horn Clauses

References

- K. Claessen and N. Smallbone
  Efficient Encodings of First-order Horn Formulas in Equational Logic
  9th IJCAR, LNCS 10900, pp. 388–404, 2018
- Y. Oi and N. Hirokawa
  Moca 0.1: Moca 0.1: A First-Order Theorem Prover for Horn Clauses
  8th IWC, p.57, 2019

Word Problem

- assumptions

$$\begin{array}{lr} \forall x. & x - 0 \to x \\ \forall x. & 0 - x \to 0 \\ \forall x, y. & \mathsf{s}(x) - \mathsf{s}(y) \to x - y \\ & \mathsf{f}(0, y) \to \mathsf{F} \\ \forall x. & \mathsf{f}(x - x, y) \to \mathsf{F} \\ \forall x. & \mathsf{f}(\mathsf{s}(x), x) \to \mathsf{T} \end{array}$$

- goal (to be refuted)

$$\mathsf{F} \approx \mathsf{T}$$

completion succeeds! $\mathsf{F} \approx \mathsf{T}$ is refuted, so $x + x \neq \mathsf{s}(x)$ for any x

Summary

1. completion
2. automation
3. Horn Clauses

thanks for your attention!