Ground joinability and connectedness in the superposition calculus

André Duarte Konstantin Korovin

andrepd@protonmail.com

8th August 2022

The University of Manchester

Introduction

Equational completion

$$\frac{l \approx r \quad s[u] \approx t}{(s[u \mapsto r] \approx t)\theta}$$

Introduction

Equational completion

$$\frac{l \approx r \quad s[u] \approx t}{(s[u \mapsto r] \approx t)\theta}$$

Resolution

$$\frac{p \vee C \quad \neg q \vee D}{(C \vee D)\theta}$$

Introduction

Equational completion

$$\frac{l \approx r \quad s[u] \approx t}{(s[u \mapsto r] \approx t)\theta}$$

Resolution

$$\frac{p \vee C \quad \neg q \vee D}{(C \vee D)\theta}$$

Superposition

$$\frac{l \approx r \vee C \quad s[u] \dot{\approx} t \vee D}{(s[u \mapsto r] \dot{\approx} t \vee C \vee D)\theta}$$

Introduction

Problem: Superposition when reduced to unit equalities is not equivalent to equational completion!

Introduction

Problem: Superposition when reduced to unit equalities is not equivalent to equational completion!

Unit Equality CNF	Waldme 710	E 2.4	Twee 2.2	Vampire 4.4
Solved/200	164/200	161/200	151/200	142/200
Solutions	164 82%	161 80%	150 75%	142 71%

Introduction

Problem: Superposition when reduced to unit equalities is not equivalent to equational completion!

Unit Equality CNF	Waldme 710	E 2.4	Twee 2.2	Vampire 4.4
Solved/200	164/200	161/200	151/200	142/200
Solutions	164 82%	161 80%	150 75%	142 71%

Best provers for UEQ (CASC/21):

- Twee (completion)
- E (superposition)
- Waldmeister (completion)

Best provers for FOF (CASC/21):

- Vampire (superposition)
- iProver (inst-gen & superposition)
- E (superposition)

Introduction

Problem: Superposition when reduced to unit equalities is not equivalent to equational completion!

Unit Equality CNF	Waldme 710	E 2.4	Twee 2.2	Vampire 4.4
Solved/200	164/200	161/200	151/200	142/200
Solutions	164 82%	161 80%	150 75%	142 71%

Best provers for UEQ (CASC/21):

- Twee (completion)
- E (superposition)
- Waldmeister (completion)

Best provers for FOF (CASC/21):

- Vampire (superposition)
- iProver (inst-gen & superposition)
- E (superposition)

Reasons (among others): ground joinability, critical pair criteria, stronger simplification.

Contributions

- Proof of completeness of superposition, wrt. closure redundancy

Contributions

- Proof of completeness of superposition, wrt. closure redundancy
- Demodulation strenghtened to encompassment demodulation (generalisation of simplfication from eq. completion)

Contributions

- Proof of completeness of superposition, wrt. closure redundancy
- Demodulation strenghtened to encompassment demodulation (generalisation of simplfication from eq. completion)
- Completeness of simplification rules:

Contributions

- Proof of completeness of superposition, wrt. closure redundancy
- Demodulation strenghtened to encompassment demodulation (generalisation of simplfication from eq. completion)
- Completeness of simplification rules:
  - Ground joinability

Contributions

- Proof of completeness of superposition, wrt. closure redundancy
- Demodulation strenghtened to encompassment demodulation (generalisation of simplfication from eq. completion)
- Completeness of simplification rules:
    - Ground joinability
    - Connectedness

Contributions

- Proof of completeness of superposition, wrt. closure redundancy
- Demodulation strenghtened to encompassment demodulation (generalisation of simplfication from eq. completion)
- Completeness of simplification rules:
  - Ground joinability
  - Connectedness
  - Ground connectedness

Contributions

- Proof of completeness of superposition, wrt. closure redundancy
- Demodulation strenghtened to encompassment demodulation (generalisation of simplfication from eq. completion)
- Completeness of simplification rules:
  - Ground joinability
  - Connectedness
  - Ground connectedness
- Practical algorithm for checking ground joinability

Refutational completeness

Redundancy:

All instances of C follow from smaller instances in S $\implies$ Clause C redundant wrt. S

Refutational completeness

Redundancy [DK21]:

All ground closures of C follow from smaller ground closures in S $\Longrightarrow$ Clause C closure redundant wrt. S

Refutational completeness

Redundancy [DK21]:

All ground closures of C follow from smaller ground closures in S $\Longrightarrow$ Clause C closure redundant wrt. S

Ground closure $= \langle$clause, grounding substitution$\rangle$.
Model construction proof is based on an ordering on ground closures.

Refutational completeness

Redundancy [DK21]:

All ground closures of C follow from smaller ground closures in S $\Longrightarrow$ Clause C closure redundant wrt. S

Ground closure $=$ $\langle$clause, grounding substitution$\rangle$.
Model construction proof is based on an ordering on ground closures.

Main idea: more general instances of terms, literals, clauses, should be smaller.

Refutational completeness

Theorem

The superposition calculus is refutationally complete wrt. closure redundancy: a set of clauses, where the conclusion of all non-redundant inferences with premises in the set is also in the set or redundant, is satisfiable.

Encompassment demodulation

Demodulation $$\frac{l \approx r \quad \cancel{C[l\theta]}}{C[l\theta \mapsto r\theta]}, \quad \begin{array}{l}\text{where } l\theta \succ r\theta, \\ \text{and } C \prec (l\theta \approx r\theta)\end{array}$$

Encompassment demodulation

$$\text{Encompassment Demodulation} \quad \frac{l \approx r \quad \cancel{C[l\theta]}}{C[l\theta \mapsto r\theta]},$$

where $l\theta \succ r\theta$, and
either C is not a positive unit
or (let $C = s[l\theta] \approx t$) $l\theta \neq s$
or $l\theta \sqsupset l$ or $s \prec t$ or $r\theta \prec t$

Encompassment demodulation

Encompassment Demodulation $\dfrac{l \approx r \quad \cancel{C[l\theta]}}{C[l\theta \mapsto r\theta]}$, where $l\theta \succ r\theta$, and either C is not a positive unit or (let $C = s[l\theta] \approx t$) $l\theta \neq s$ or $l\theta \sqsupset l$ or $s \prec t$ or $r\theta \prec t$

Encompassment demodulation

$$\text{Encompassment Demodulation}\quad \frac{l \approx r \quad \cancel{C[l\theta]}}{C[l\theta \mapsto r\theta]},$$

where $l\theta \succ r\theta$, and
either C is not a positive unit
or (let $C = s[l\theta] \approx t$) $l\theta \neq s$
or $l\theta \sqsupset l$ or $s \prec t$ or $r\theta \prec t$

Strict generalisation of "simplify with oriented rule" of eq. completion!

Encompassment demodulation

$$\text{Encompassment Demodulation} \quad \frac{l \approx r \quad \cancel{C[l\theta]}}{C[l\theta \mapsto r\theta]}, \quad \begin{array}{l} \text{where } l\theta \succ r\theta, \text{ and} \\ \text{either } C \text{ is not a positive unit} \\ \text{or (let } C = s[l\theta] \approx t) \; l\theta \neq s \\ \text{or } l\theta \sqsupset l \text{ or } s \prec t \text{ or } r\theta \prec t \end{array}$$

Strict generalisation of "simplify with oriented rule" of eq. completion!

Examples:

- $f(x) \approx b$ can rewrite $f(a) \approx c$, even if $b \succ c$
- $f(a) \approx b$ can rewrite $(f(a) \approx c \vee f(a) \approx d)$, even if $b \succ c$ and $b \succ d$

Encompassment demodulation

$$\text{Encompassment Demodulation} \quad \frac{l \approx r \quad \cancel{C[l\theta]}}{C[l\theta \mapsto r\theta]},$$

where $l\theta \succ r\theta$, and
either C is not a positive unit
or (let $C = s[l\theta] \approx t$) $l\theta \neq s$
or $l\theta \sqsupset l$ or $s \prec t$ or $r\theta \prec t$

Strict generalisation of "simplify with oriented rule" of eq. completion!

Examples:

- $f(x) \approx b$ can rewrite $f(a) \approx c$, even if $b \succ c$
- $f(a) \approx b$ can rewrite $(f(a) \approx c \vee f(a) \approx d)$, even if $b \succ c$ and $b \succ d$

Neither are allowed in "usual" demodulation.

Encompassment demodulation

$$\text{Encompassment Demodulation}\quad \frac{l \approx r \quad \cancel{C[l\theta]}}{C[l\theta \mapsto r\theta]},$$

where $l\theta \succ r\theta$, and
either C is not a positive unit
or (let $C = s[l\theta] \approx t$) $l\theta \neq s$
or $l\theta \sqsupset l$ or $s \prec t$ or $r\theta \prec t$

Strict generalisation of "simplify with oriented rule" of eq. completion!

Examples:

- $f(x) \approx b$ can rewrite $f(a) \approx c$, even if $b \succ c$
- $f(a) \approx b$ can rewrite $(f(a) \approx c \vee f(a) \approx d)$, even if $b \succ c$ and $b \succ d$

Neither are allowed in "usual" demodulation.

(Also faster to check)

Encompassment demodulation

Theorem

Encompassment demodulation is a closure redundancy of the superposition calculus.

Ground joinability

In equational completion:

$$E \cup \{s \approx t\} \vdash E, \quad \text{if all ground } s\sigma \downarrow_{\succ E} t\sigma$$

($\downarrow_{\succ E}$: joinable using smaller equations in E, for definition see [MN90])

Ground joinability

In equational completion:

$$E \cup \{s \approx t\} \vdash E, \quad \text{if all ground } s\sigma \downarrow_{\succ E} t\sigma$$

($\downarrow_{\succ E}$: joinable using smaller equations in E, for definition see [MN90])

Example:

- $\{xy \approx yx,\ (xy)z \approx x(yz)\} \cup \cancel{\{x(yz) \approx z(xy)\}}$

Ground joinability

In equational completion:

$$E \cup \{s \approx t\} \vdash E, \quad \text{if all ground } s\sigma \downarrow_{\succ E} t\sigma$$

($\downarrow_{\succ E}$: joinable using smaller equations in E, for definition see [MN90])

Example:

- $\{xy \approx yx,\ (xy)z \approx x(yz)\} \cup \cancel{\{x(yz) \approx z(xy)\}}$

The proofs of correctness rely on the proof orderings technique.

Ground joinability

In equational completion:

$$E \cup \{s \approx t\} \vdash E, \quad \text{if all ground } s\sigma \downarrow_{\succ E} t\sigma$$

($\downarrow_{\succ E}$: joinable using smaller equations in E, for definition see [MN90])

Example:

- $\{xy \approx yx,\ (xy)z \approx x(yz)\} \cup \{\cancel{x(yz) \approx z(xy)}\}$

The proofs of correctness rely on the proof orderings technique.
$\Longrightarrow$ Ground joinability wasn't available for superposition provers.

Ground joinability

Ground joinability

$$\frac{\cancel{s \approx t \vee C} \quad S}{}, \quad \text{where all } s\sigma \wr t\sigma \text{ in } s \approx t \vee C \text{ wrt. } S$$

$$\frac{\cancel{s \not\approx t \vee C} \quad S}{C}, \quad \text{where all } s\sigma \wr t\sigma \text{ in } s \not\approx t \vee C \text{ wrt. } S$$

($\wr$: joinable using smaller equations in S, for definition see [DK22])

Generalisation of ground completeness for full clauses

Ground joinability

Theorem

Ground joinability is a closure redundancy of the superposition calculus. (In superposition, deleting/simplifying clauses with ground joinable literals does not compromise refutational completeness.)

Algorithm for ground joinability

If two (non-ground) terms are joinable under every preorder among their variables, then they are ground joinable.

Naïve algorithm is $\mathcal{O}(n!e^n)$.

Algorithm for ground joinability

If two (non-ground) terms are joinable under every preorder among their variables, then they are ground joinable.

Naïve algorithm is $\mathcal{O}(n!e^n)$.

Joinability under a partial $\succeq_v \implies$ joinability under all total $\succeq_v' \supseteq \succeq_v$.

Algorithm for ground joinability

If two (non-ground) terms are joinable under every preorder among their variables, then they are ground joinable.

Naïve algorithm is $\mathcal{O}(n!e^n)$.

Joinability under a partial $\succeq_v \implies$ joinability under all total $\succeq'_v \supseteq \succeq_v$.

So the question is to choose which preorders to check, and when to stop.

Algorithm for ground joinability

If two (non-ground) terms are joinable under every preorder among their variables, then they are ground joinable.

Naïve algorithm is $\mathcal{O}(n!e^n)$.

Joinability under a partial $\succeq_v \implies$ joinability under all total $\succeq'_v \supseteq \succeq_v$.

So the question is to choose which preorders to check, and when to stop.

Goals:

- If terms are ground joinable: conclude this with the least number of preorders possible
- If terms are not ground joinable: find a preorder to disprove it as soon as possible

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Start with base ordering

  $z(x(yx)) \approx z(xy)$ $\quad\quad$ $\succ = \succ_t$ $\quad\quad$ Queue:

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Reduce

  $z(x(yx)) \approx z(xy)$ $\quad\quad \succ = \succ_t \quad\quad$ Queue:

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Try to rewrite with extension

  $z(x(yx)) \approx z(xy)$ $\quad\quad$ $\succ = \succ_t \cup \{x \prec y\}$ $\quad\quad$ Queue:

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Try to rewrite with extension

  $z(x(xy)) \approx z(xy)$ $\quad \succ = \succ_t \cup \{x \prec y\}$ $\quad$ Queue:

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Reduce

  $z(x(xy)) \approx z(xy)$ $\qquad \succ = \succ_t \cup \{x \prec y\}$ $\qquad$ Queue:

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Reduce

  $z(xy) \approx z(xy)$

  $\succ = \succ_t \cup \{x \prec y\}$

  Queue:

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Same normal form: add orders not covered to queue

  $z(xy) \approx z(xy)$ $\quad \succ = \succ_t \cup \{x \prec y\}$

  Queue:
  $x \sim y$
  $x \succ y$

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Get ordering from queue

  $z(x(yx)) \approx z(xy)$ $\quad\quad \succ = \succ_t \cup \{x \succ y\}$ $\quad\quad$ Queue: $x \sim y$

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Reduce

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

  $\succ = \succ_t \cup \{x \succ y\}$

  Queue:

  $x \sim y$

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Reduce

  $z(x(yx)) \approx z(yx)$

  $\succ = \succ_t \cup \{x \succ y\}$

  Queue:
  $x \sim y$

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Reduce

  $z(x(yx)) \approx z(yx)$

  $\succ = \succ_t \cup \{x \succ y\}$

  Queue:

  $x \sim y$

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Reduce

  $z(y(xx)) \approx z(yx)$

  $\succ = \succ_t \cup \{x \succ y\}$

  Queue:
  $x \sim y$

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Reduce

  $z(y(xx)) \approx z(yx)$

  $\succ = \succ_t \cup \{x \succ y\}$

  Queue:

  $x \sim y$

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Same normal form: add orders not covered to queue

$z(yx) \approx z(yx)$	$\succ = \succ_t$	Queue:
	$\cup \{x \succ y\}$	$x \sim y$

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Get ordering from queue

  $z(x(yx)) \approx z(xy)$ $\quad\quad \succ = \succ_t \cup \{x \sim y\}$ $\quad\quad$ Queue:

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Reduce

  $z(x(xx)) \approx z(xx)$ $\qquad \succ = \succ_t \cup \{x \sim y\}$ $\qquad$ Queue:

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Reduce

  $z(x(xx)) \approx z(xx)$ $\quad\quad \succ = \succ_t \cup \{x \sim y\}$ $\quad\quad$ Queue:

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Reduce

  $z(xx) \approx z(xx)$

  $\succ \,=\, \succ_t$
  $\cup \{x \sim y\}$

  Queue:

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Same normal form: add orders not covered to queue

  $z(xx) \approx z(xx)$ $\quad \succ = \succ_t \cup \{x \sim y\}$ $\quad$ Queue:

Algorithm for ground joinability

Example: $S = \{xy \approx yx,\ x(yz) \approx y(xz),\ xx \approx x,\ x(xy) \approx xy\}$

- $z(x(yx)) \approx z(xy)$

  Queue empty: ground joinable

  $z(xx) \approx z(xx)$ $\succ = \succ_t \cup \{x \sim y\}$ Queue:

Connectedness

Critical pair criteria. In equational completion, if $s \leftarrow u \rightarrow t$ [BD88]:

$$E \cup \{s \approx t\} \vdash E, \quad \begin{array}{l} \text{if } s \leftrightarrow_E v_1 \leftrightarrow_E \cdots \leftrightarrow_E v_n \leftrightarrow_E t \\ \text{with } v_1, \ldots, v_n \prec u \end{array}$$

i.e. we can rewrite in *opposite* direction ($\succ$) if still smaller than u.

Connectedness

Critical pair criteria. In equational completion, if $s \leftarrow u \rightarrow t$ [BD88]:

$$E \cup \{s \approx t\} \vdash E, \quad \begin{array}{l}\text{if } s \leftrightarrow_E v_1 \leftrightarrow_E \cdots \leftrightarrow_E v_n \leftrightarrow_E t \\ \text{with } v_1, \ldots, v_n \prec u\end{array}$$

i.e. we can rewrite in *opposite* direction ($\succ$) if still smaller than u.

Example:

- $S = \{x + y \approx y + x,\ x + (y + \overline{x}) \approx 1,\ x \cdot (x + y) \approx x,\ \ldots\}$
  $x + (\overline{x} + y) \leftarrow x + (\overline{x} \cdot y) \rightarrow 1$

Connectedness

Critical pair criteria. In equational completion, if $s \leftarrow u \rightarrow t$ [BD88]:

$$E \cup \{s \approx t\} \vdash E, \quad \begin{array}{l}\text{if } s \leftrightarrow_E v_1 \leftrightarrow_E \cdots \leftrightarrow_E v_n \leftrightarrow_E t \\ \text{with } v_1, \ldots, v_n \prec u\end{array}$$

i.e. we can rewrite in *opposite* direction ($\succ$) if still smaller than u.

Example:

- $S = \{x + y \approx y + x,\ x + (y + \overline{x}) \approx 1,\ x \cdot (x + y) \approx x,\ \ldots\}$
  $x + (\overline{x} + y) \leftarrow x + (\overline{x} \cdot y) \rightarrow 1$

$$x + (\overline{x} + y) \xleftrightarrow{x+y \approx y+x} x + (y + \overline{x}) \xleftrightarrow{x+(y+\overline{x}) \approx 1} 1$$

Connectedness

Critical pair criteria. In equational completion, if $s \leftarrow u \rightarrow t$ [BD88]:

$$E \cup \{s \approx t\} \vdash E, \quad \begin{array}{l}\text{if } s \leftrightarrow_E v_1 \leftrightarrow_E \cdots \leftrightarrow_E v_n \leftrightarrow_E t \\ \text{with } v_1, \ldots, v_n \prec u\end{array}$$

i.e. we can rewrite in *opposite* direction ($\succ$) if still smaller than u.

Example:

- $S = \{x + y \approx y + x,\ x + (y + \overline{x}) \approx 1,\ x \cdot (x + y) \approx x,\ \ldots\}$
  $x + (\overline{x} + y) \leftarrow x + (\overline{x} \cdot y) \rightarrow 1$

$$x + (\overline{x} + y) \xleftrightarrow{x+y\approx y+x} x + (y + \overline{x}) \xleftrightarrow{x+(y+\overline{x})\approx 1} 1$$

Connectedness

In superposition a analogous criteria exists. Generating inferences

$$\text{Superposition} \qquad \frac{l \approx r \vee C \quad s[u] \dot{\approx} t \vee D}{(s[u \mapsto r] \dot{\approx} t \vee C \vee D)\rho}$$

where $s[u \mapsto r]\rho$ and $t\rho$ are connected under $\{l \approx r \vee C,\ s \approx t \vee D\}$ and unifier ρ wrt. some set of clauses S, are redundant inferences wrt. S.
(connected: generalisation of criterion for completion, for definition see [DK22])

Connectedness

In superposition a analogous criteria exists. Generating inferences

$$\text{Superposition} \qquad \frac{l \approx r \vee C \quad s[u] \dot{\approx} t \vee D}{(s[u \mapsto r] \dot{\approx} t \vee C \vee D)\rho}$$

where $s[u \mapsto r]\rho$ and $t\rho$ are connected under $\{l \approx r \vee C,\ s \approx t \vee D\}$ and unifier ρ wrt. some set of clauses S, are redundant inferences wrt. S.
(connected: generalisation of criterion for completion, for definition see [DK22])

Redundant clause: can be deleted in any context & used for simplifications
Conclusion of redundant inference: can be deleted in that context only.

Connectedness

Theorem

Connectedness is a closure redundancy of the superposition calculus. (In superposition, inferences where the conclusion is connected under the premises can be skipped without compromising refutational completeness.)

Summary

- Encompassment demodulation is a redundancy of superposition
  - Superposition calculus with encomp. demod. and tautology deletion is equivalent to unfailing completion on unit equations
- Ground joinability is a redundancy of superposition
- Connectedness is a redundancy of superposition
- Implementing ground joinability in iProver (using our algorithm) increases the overall number of problems solved and solves hitherto unsolved problems

References

[BD88] Leo Bachmair and Nachum Dershowitz. "Critical Pair Criteria for Completion". In: *Journal of Symbolic Computation* (1988).

[MN90] Ursula Martin and Tobias Nipkow. "Ordered rewriting and confluence". In: *10th International Conference on Automated Deduction*. 1990.

[DK21] André Duarte and Konstantin Korovin. "AC Simplifications and Closure Redundancies in the Superposition Calculus". In: *Automated Reasoning with Analytic Tableaux and Related Methods – 30th International Conference, TABLEAUX, Proceedings*. 2021.

[DK22] André Duarte and Konstantin Korovin. "Ground Joinability and Connectedness in the Superposition Calculus". In: *Automated Reasoning – 11th International Joint Conference, IJCAR, Proceedings*. 2022.