

Automated Deduction

Laura Kovács

TU Wien

Outline

Saturation Algorithms (contd)

Redundancy Elimination

Inference Process

Inference process: sequence of sets of formulas S_0, S_1, \dots , denoted by

$$S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \dots$$

$(S_i \Rightarrow S_{i+1})$ is a **step** of this process.

Inference Process

Inference process: sequence of sets of formulas S_0, S_1, \dots , denoted by

$$S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \dots$$

$(S_i \Rightarrow S_{i+1})$ is a **step** of this process.

We say that this step is an **I-step** if

1. there exists an inference

$$\frac{F_1 \quad \dots \quad F_n}{F}$$

in \mathbb{I} such that $\{F_1, \dots, F_n\} \subseteq S_i$;

2. $S_{i+1} = S_i \cup \{F\}$.

Inference Process

Inference process: sequence of sets of formulas S_0, S_1, \dots , denoted by

$$S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \dots$$

$(S_i \Rightarrow S_{i+1})$ is a **step** of this process.

We say that this step is an **I-step** if

1. there exists an inference

$$\frac{F_1 \quad \dots \quad F_n}{F}$$

in \mathbb{I} such that $\{F_1, \dots, F_n\} \subseteq S_i$;

2. $S_{i+1} = S_i \cup \{F\}$.

An **I-inference process** is an inference process whose every step is an I-step.

Property

Let $S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \dots$ be an \mathbb{I} -inference process and a formula F belongs to some S_i . Then S_i is derivable in \mathbb{I} from S_0 . In particular, every S_i is a subset of the \mathbb{I} -closure of S_0 .

Limit of a Process

The **limit** of an inference process $S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \dots$ is the set of formulas $\bigcup_i S_i$.

Limit of a Process

The **limit** of an inference process $S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \dots$ is the set of formulas $\bigcup_i S_i$.

In other words, the limit is **the set of all derived formulas**.

Limit of a Process

The **limit** of an inference process $S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \dots$ is the set of formulas $\bigcup_i S_i$.

In other words, the limit is the set of all derived formulas.

Suppose that we have an infinite inference process such that S_0 is **unsatisfiable** and we use the **binary resolution inference system**.

Limit of a Process

The **limit** of an inference process $S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \dots$ is the set of formulas $\bigcup_i S_i$.

In other words, the limit is the set of all derived formulas.

Suppose that we have an infinite inference process such that S_0 is **unsatisfiable** and we use the **binary resolution inference system**.

Question: does completeness imply that the limit of the process contains the empty clause?

Fairness

Let $S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \dots$ be an inference process with the limit S_ω .
The process is called **fair** if for every \mathbb{I} -inference

$$\frac{F_1 \quad \dots \quad F_n}{F},$$

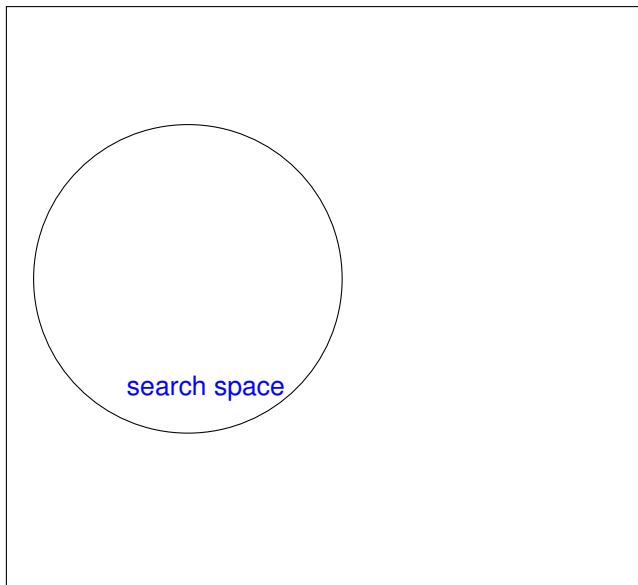
if $\{F_1, \dots, F_n\} \subseteq S_\omega$, then there exists i such that $F \in S_i$.

Completeness, reformulated

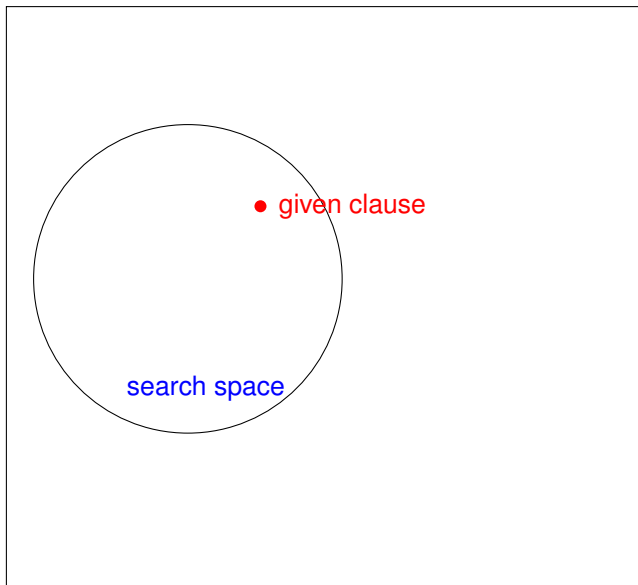
Theorem Let \mathbb{I} be an inference system. The following conditions are equivalent.

1. \mathbb{I} is complete.
2. For every unsatisfiable set of formulas S_0 and any fair \mathbb{I} -inference process with the initial set S_0 , the limit of this inference process contains \square .

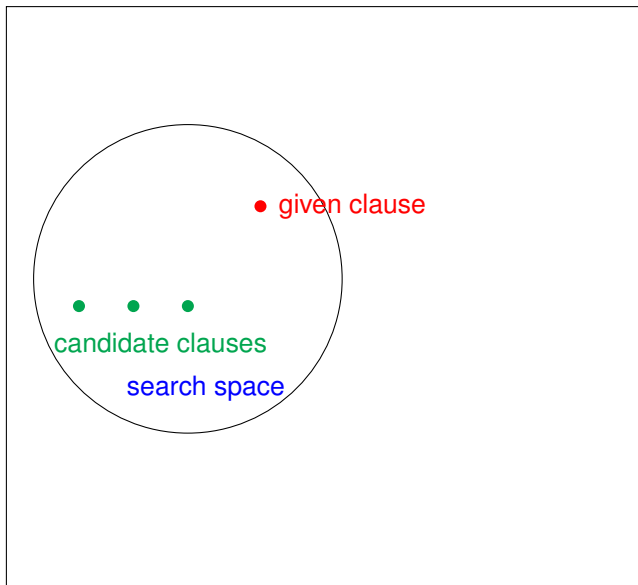
Fair Saturation Algorithms: Inference Selection by Clause Selection



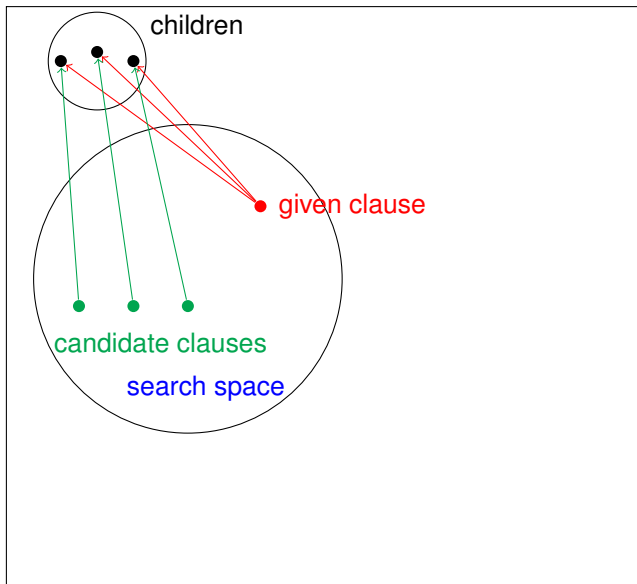
Fair Saturation Algorithms: Inference Selection by Clause Selection



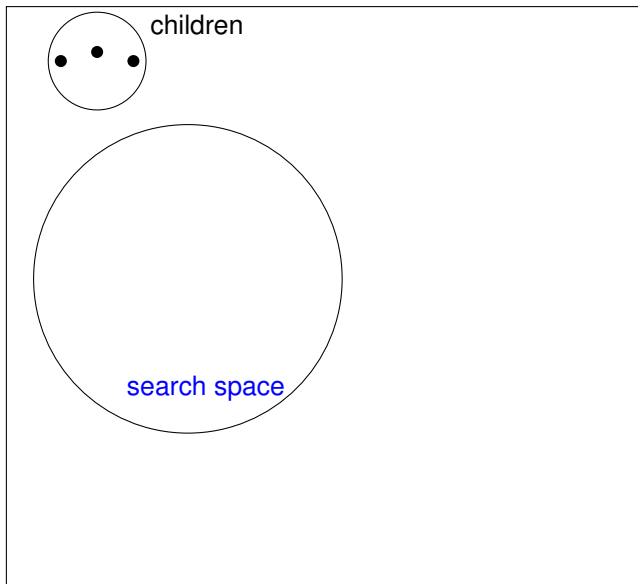
Fair Saturation Algorithms: Inference Selection by Clause Selection



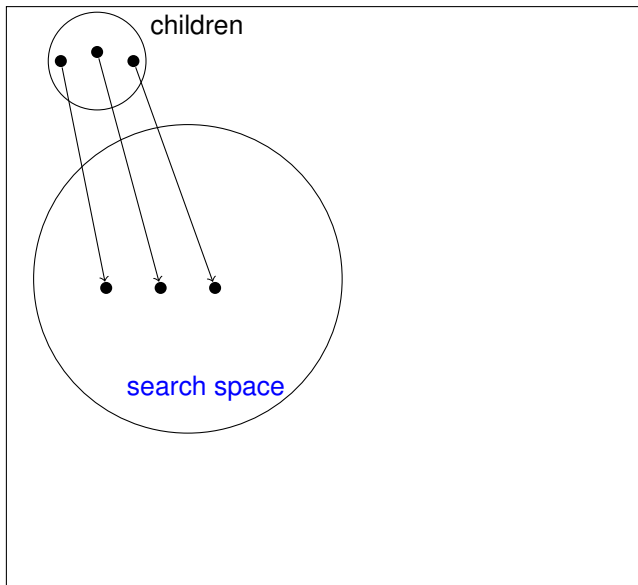
Fair Saturation Algorithms: Inference Selection by Clause Selection



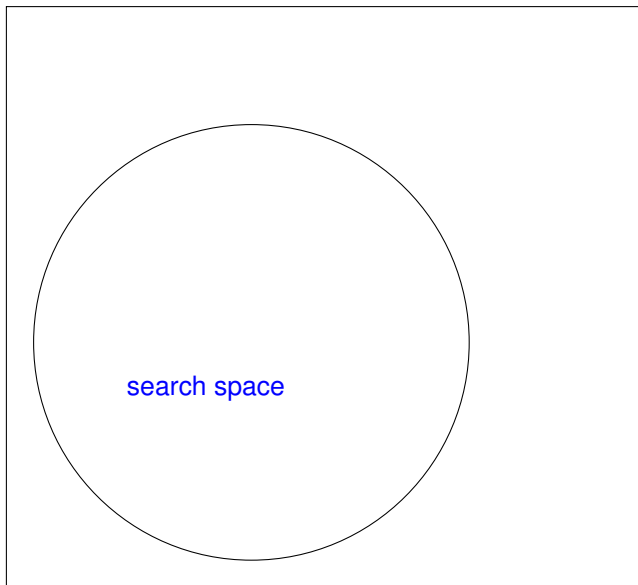
Fair Saturation Algorithms: Inference Selection by Clause Selection



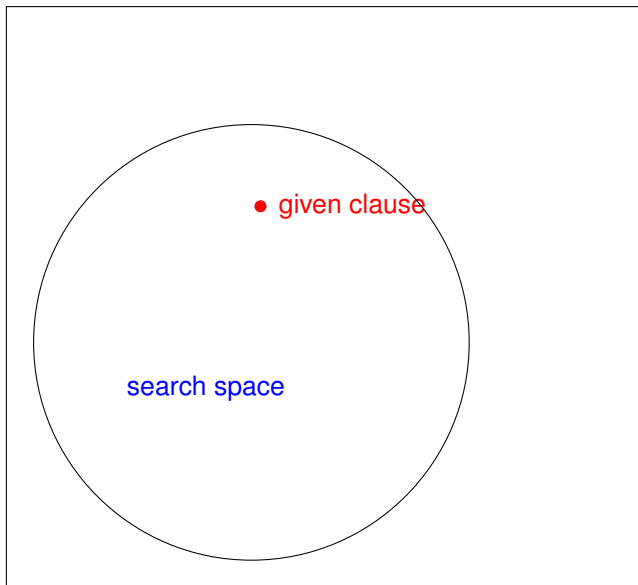
Fair Saturation Algorithms: Inference Selection by Clause Selection



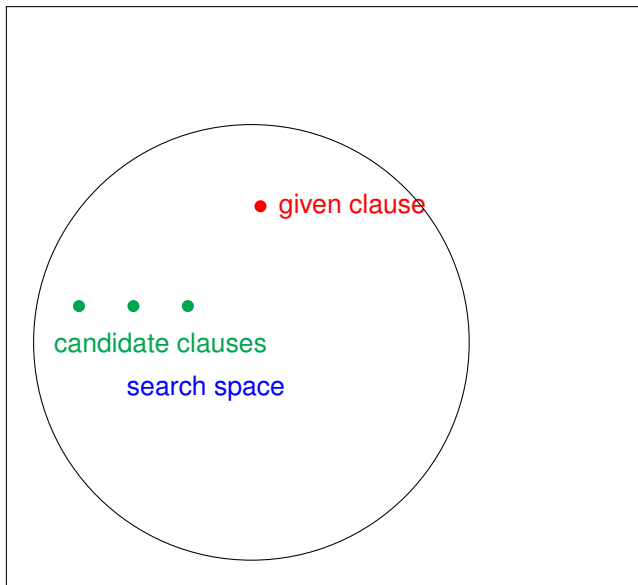
Fair Saturation Algorithms: Inference Selection by Clause Selection



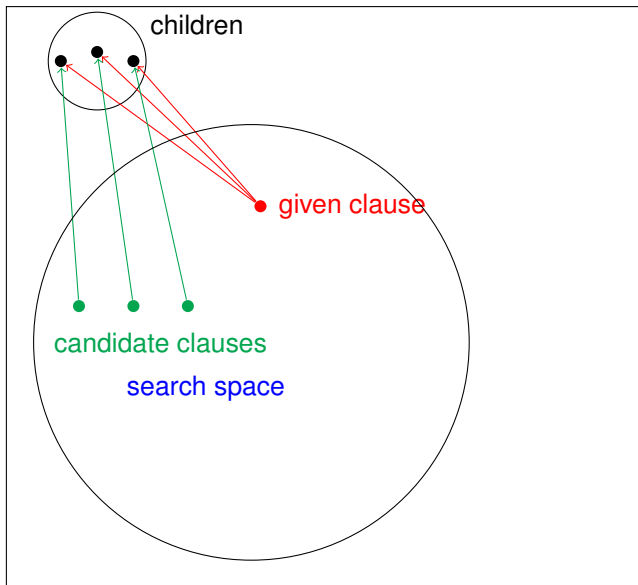
Fair Saturation Algorithms: Inference Selection by Clause Selection



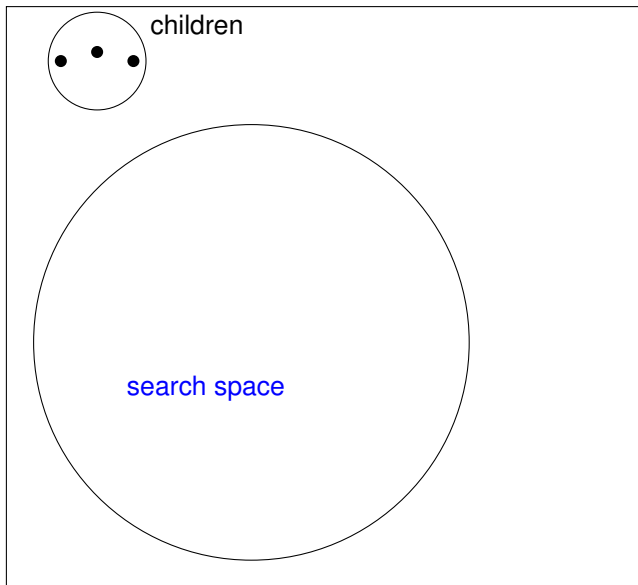
Fair Saturation Algorithms: Inference Selection by Clause Selection



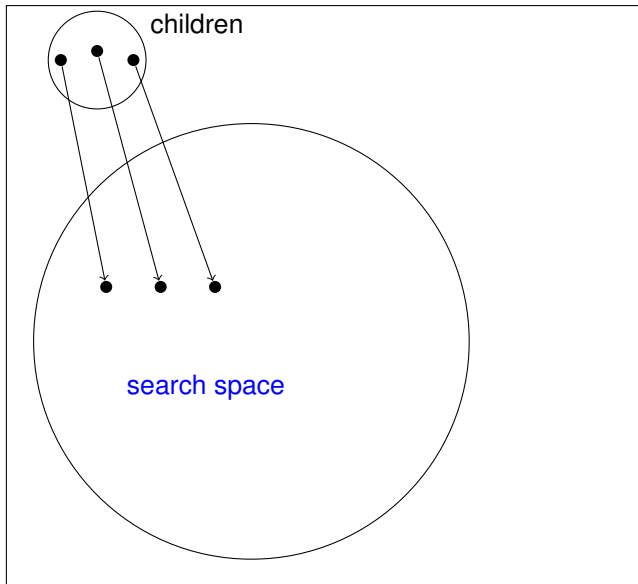
Fair Saturation Algorithms: Inference Selection by Clause Selection



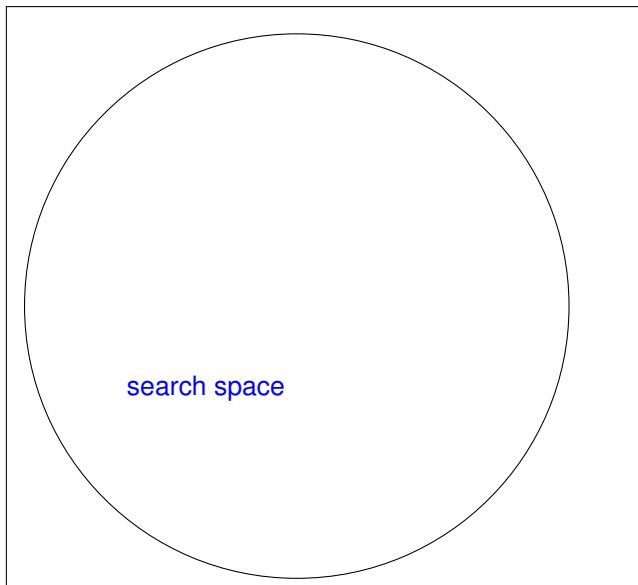
Fair Saturation Algorithms: Inference Selection by Clause Selection



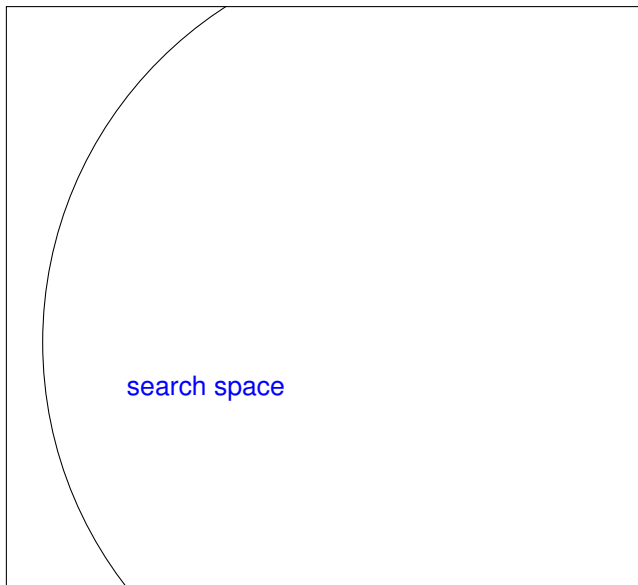
Fair Saturation Algorithms: Inference Selection by Clause Selection



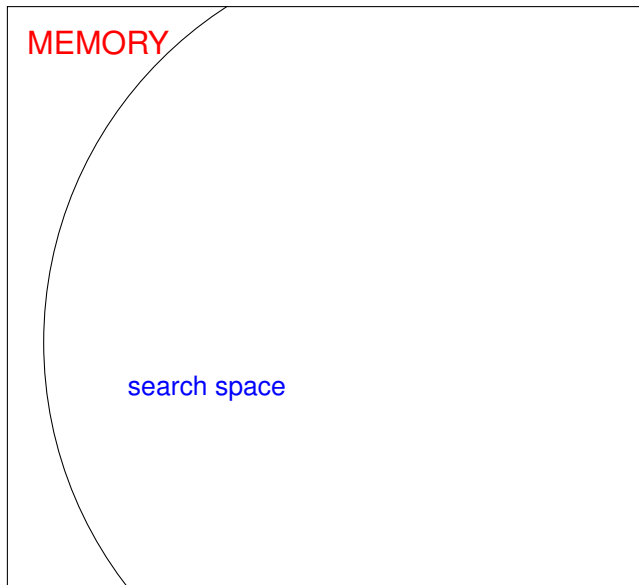
Fair Saturation Algorithms: Inference Selection by Clause Selection



Fair Saturation Algorithms: Inference Selection by Clause Selection



Fair Saturation Algorithms: Inference Selection by Clause Selection



Saturation Algorithm

A **saturation algorithm** tries to **saturate** a set of clauses with respect to a given inference system.

In theory there are three possible scenarios:

1. At some moment the empty clause \square is generated, in this case the input set of clauses is unsatisfiable.
2. Saturation will terminate without ever generating \square , in this case the input set of clauses is satisfiable.
3. Saturation will run **forever**, but without generating \square . In this case the input set of clauses is satisfiable.

Saturation Algorithm in Practice

In practice there are three possible scenarios:

1. At some moment the empty clause \square is generated, in this case the input set of clauses is unsatisfiable.
2. Saturation will terminate without ever generating \square , in this case the input set of clauses is satisfiable.
3. Saturation will run until we run out of resources, but without generating \square . In this case it is unknown whether the input set is unsatisfiable.

Outline

Saturation Algorithms (contd)

Redundancy Elimination

Subsumption and Tautology Deletion

A clause is a propositional tautology if it is of the form $p \vee \neg p \vee C$, that is, it contains a pair of complementary literals. There are also **equational tautologies**, for example $a \neq b \vee b \neq c \vee f(c, c) = f(a, a)$.

Subsumption and Tautology Deletion

A clause is a propositional tautology if it is of the form $p \vee \neg p \vee C$, that is, it contains a pair of complementary literals.

There are also **equational tautologies**, for example $a \neq b \vee b \neq c \vee f(c, c) = f(a, a)$.

A clause C **subsumes** any clause $C \vee D$, where D is non-empty.

Subsumption and Tautology Deletion

A clause is a propositional tautology if it is of the form $p \vee \neg p \vee C$, that is, it contains a pair of complementary literals.

There are also **equational tautologies**, for example $a \neq b \vee b \neq c \vee f(c, c) = f(a, a)$.

A clause C **subsumes** any clause $C \vee D$, where D is non-empty.

It was known since 1965 that **subsumed clauses and propositional tautologies can be removed from the search space.**

Problem

How can we **prove** that **completeness is preserved** if we **remove subsumed clauses and tautologies** from the **search space**?

Problem

How can we **prove** that **completeness is preserved** if we **remove subsumed clauses and tautologies** from the **search space**?

Solution: general **theory of redundancy**.

Bag Extension of an Ordering

Bag = finite multiset.

Let $>$ be any (strict) ordering on a set X . The **bag extension of $>$** is a binary relation $>^{bag}$, on bags over X , defined as the smallest transitive relation on bags such that

$$\{x, y_1, \dots, y_n\} >^{bag} \{x_1, \dots, x_m, y_1, \dots, y_n\}$$

if $x > x_i$ for all $i \in \{1 \dots m\}$,

where $m \geq 0$.

Bag Extension of an Ordering

Bag = finite multiset.

Let $>$ be any (strict) ordering on a set X . The **bag extension of $>$** is a binary relation $>^{bag}$, on bags over X , defined as the smallest transitive relation on bags such that

$$\{x, y_1, \dots, y_n\} >^{bag} \{x_1, \dots, x_m, y_1, \dots, y_n\}$$

if $x > x_i$ for all $i \in \{1 \dots m\}$,

where $m \geq 0$.

Idea: a bag becomes smaller if we replace an element by **any finite number** of smaller elements.

Bag Extension of an Ordering

Bag = finite multiset.

Let $>$ be any (strict) ordering on a set X . The **bag extension** of $>$ is a binary relation $>^{bag}$, on bags over X , defined as the smallest transitive relation on bags such that

$$\{x, y_1, \dots, y_n\} >^{bag} \{x_1, \dots, x_m, y_1, \dots, y_n\} \\ \text{if } x > x_i \text{ for all } i \in \{1 \dots m\},$$

where $m \geq 0$.

Idea: a bag becomes smaller if we replace an element by **any finite number** of smaller elements.

The following **results are known** about the bag extensions of orderings:

1. $>^{bag}$ is an **ordering**;
2. If $>$ is **total**, then so is $>^{bag}$;
3. If $>$ is **well-founded**, then so is $>^{bag}$.

Clause Orderings

From now on consider clauses also as **bags of literals**. Note:

- ▶ we have an ordering \succ for comparing literals;
- ▶ a clause is a bag of literals.

Clause Orderings

From now on consider clauses also as **bags of literals**. Note:

- ▶ we have an ordering \succ for comparing literals;
- ▶ a clause is a bag of literals.

Hence

- ▶ we can compare clauses using the **bag extension** \succ^{bag} of \succ .

Clause Orderings

From now on consider clauses also as **bags of literals**. Note:

- ▶ we have an ordering \succ for comparing literals;
- ▶ a clause is a bag of literals.

Hence

- ▶ we can compare clauses using the **bag extension** \succ^{bag} of \succ .

For simplicity we denote the multiset ordering also by \succ .

Redundancy

A clause $C \in S$ is called **redundant in S** if it is a logical consequence of clauses in S strictly smaller than C .

Examples

A **tautology** $p \vee \neg p \vee C$ is a logical consequence of the empty set of formulas:

$$\models p \vee \neg p \vee C,$$

therefore it is **redundant**.

Examples

A **tautology** $p \vee \neg p \vee C$ is a logical consequence of the empty set of formulas:

$$\models p \vee \neg p \vee C,$$

therefore it is **redundant**.

We know that C **subsumes** $C \vee D$. Note

$$\begin{aligned} C \vee D &\succ C \\ C &\models C \vee D \end{aligned}$$

therefore subsumed clauses are **redundant**.

Examples

A **tautology** $p \vee \neg p \vee C$ is a logical consequence of the empty set of formulas:

$$\models p \vee \neg p \vee C,$$

therefore it is **redundant**.

We know that C **subsumes** $C \vee D$. Note

$$\begin{aligned} C \vee D &\succ C \\ C &\models C \vee D \end{aligned}$$

therefore subsumed clauses are **redundant**.

If $\square \in S$, then all non-empty other clauses in S are **redundant**.

Redundant Clauses Can be Removed

In \mathbb{BR}_σ (and in all calculi we will consider later) **redundant clauses can be removed from the search space.**

Redundant Clauses Can be Removed

In BR_σ (and in all calculi we will consider later) **redundant clauses can be removed from the search space.**

Inference Process with Redundancy

Let \mathbb{I} be an inference system. Consider an inference process with two kinds of step $S_i \Rightarrow S_{i+1}$:

1. Adding the conclusion of an \mathbb{I} -inference with premises in S_i .
2. Deletion of a clause redundant in S_i , that is

$$S_{i+1} = S_i - \{C\},$$

where C is redundant in S_i .

Fairness: Persistent Clauses and Limit

Consider an inference process

$$S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \dots$$

A clause C is called **persistent** if

$$\exists \forall j \geq i (C \in S_j).$$

The **limit** S_ω of the inference process is the set of all persistent clauses:

$$S_\omega = \bigcup_{i=0,1,\dots} \bigcap_{j \geq i} S_j.$$

Fairness

The process is called \mathbb{I} -fair if every inference with persistent premises in S_ω has been applied, that is, if

$$\frac{C_1 \quad \dots \quad C_n}{C}$$

is an inference in \mathbb{I} and $\{C_1, \dots, C_n\} \subseteq S_\omega$, then $C \in S_i$ for some i .

Completeness of BR_σ

Completeness Theorem. Let \succ be a well-founded ordering and σ a well-behaved selection function. Let also

1. \mathcal{S}_0 be a set of clauses;
2. $\mathcal{S}_0 \Rightarrow \mathcal{S}_1 \Rightarrow \mathcal{S}_2 \Rightarrow \dots$ be a fair BR_σ -inference process.

Then \mathcal{S}_0 is unsatisfiable if and only if $\square \in \mathcal{S}_i$ for some i .

Saturation up to Redundancy

A set S of clauses is called **saturated up to redundancy** if for every \mathbb{I} -inference

$$\frac{C_1 \quad \dots \quad C_n}{C}$$

with premises in S , either

1. $C \in S$; or
2. C is redundant w.r.t. S , that is, $S_{\setminus C} \models C$.

Saturation up to Redundancy and Satisfiability Checking

Lemma. A set S of clauses saturated up to redundancy is unsatisfiable if and only if $\square \in S$.

Saturation up to Redundancy and Satisfiability Checking

Lemma. A set S of clauses saturated up to redundancy is unsatisfiable if and only if $\square \in S$.

Therefore, if we built a set saturated up to redundancy, then the initial set S_0 is **satisfiable**. This is a powerful way of checking redundancy: one can even check satisfiability of formulas having only **infinite models**.

Saturation up to Redundancy and Satisfiability Checking

Lemma. A set S of clauses saturated up to redundancy is unsatisfiable if and only if $\square \in S$.

Therefore, if we built a set saturated up to redundancy, then the initial set S_0 is **satisfiable**. This is a powerful way of checking redundancy: one can even check satisfiability of formulas having only **infinite models**.

The only problem with this characterisation is that there is **no obvious way to build a model of S_0** out of a saturated set.

Binary Resolution with Selection

One of the **key properties** to satisfy this lemma is the following: the conclusion of every rule is strictly smaller than the rightmost premise of this rule.

- ▶ Binary resolution,

$$\frac{\underline{p \vee C_1} \quad \underline{\neg p \vee C_2}}{C_1 \vee C_2} \text{ (BR).}$$

- ▶ Positive factoring,

$$\frac{\underline{p \vee p \vee C}}{p \vee C} \text{ (Fact).}$$