# Automated Deduction

Laura Kovács

# Outline

Inference Systems with Selection (Recap)

Saturation Algorithms

Redundancy Elimination

# Inference Systems - Soundness (Recap)

- An inference is sound if the conclusion of this inference is a logical consequence of its premises.
- An inference system is sound if every inference rule in this system is sound.

# Inference Systems - Soundness (Recap)

$$\frac{F_1 \quad F_2}{G}$$ soundness: $F_1 \wedge F_2 \Rightarrow G$

- An inference is sound if the conclusion of this inference is a logical consequence of its premises.
- An inference system is sound if every inference rule in this system is sound.

$\mathbb{BR}$ is sound.

Consequence of soundness: let $S$ be a set of clauses. If $\square$ can be derived from $S$ in $\mathbb{BR}$, then $S$ is unsatisfiable.

Lecture 2 - Exercise recap : infinite number of $\mathbb{BR}$ derivations of $\square$

# Can this be used for checking (un)satisfiability (Recap)

1. What happens when the empty clause cannot be derived from $S$?
2. How can one search for possible derivations of the empty clause?

# Can this be used for checking (un)satisfiability (Recap)

Refutational

1. Completeness.
   *Let $S$ be an unsatisfiable set of clauses. Then there exists a derivation of $\Box$ from $S$ in $\mathbb{BR}$.*

# Can this be used for checking (un)satisfiability (Recap)

1. Completeness.
   *Let $S$ be an unsatisfiable set of clauses. Then there exists a derivation of $\square$ from $S$ in $\mathbb{BR}$.*
2. We have to formalize search for derivations.

.

# Can this be used for checking (un)satisfiability (Recap)

1. Completeness.
   *Let $S$ be an unsatisfiable set of clauses. Then there exists a derivation of $\Box$ from $S$ in $\mathbb{BR}$.*
2. We have to formalize search for derivations.

We introduced well-behaved selection functions for selecting literals in clauses and applying inferences only over selected literals.

$\underline{P} \vee Q$ $\qquad\qquad\qquad$ $\underline{\neg P} \vee \neg Q$

$Q \vee \neg Q$

# Can this be used for checking (un)satisfiability (Recap)

1. Completeness.
   *Let $S$ be an unsatisfiable set of clauses. Then there exists a derivation of $\square$ from $S$ in $\mathbb{BR}$.*
2. We have to formalize search for derivations.

We introduced well-behaved selection functions for selecting literals in clauses and applying inferences only over selected literals.

Binary resolution $\mathbb{BR}$ with selection is complete for every well-behaved selection function.

Theory : BR sound & complete

\+ selection $\sigma$

\+ saturation (fairness)

\+ redundancy

\+ equality

ATP (theory & practice) : $BR_\sigma$ is sound & complete

search for proofs

# Outline

Inference Systems with Selection (Recap)

Saturation Algorithms

Redundancy Elimination

# How to Establish Unsatisfiability?

Completeness is formulated in terms of derivability of the empty clause $\square$ from a set $S_0$ of clauses in an inference system $\mathbb{I}$. However, this formulations gives no hint on how to search for such a derivation.

# How to Establish Unsatisfiability?

$S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \ldots \Rightarrow \square$

$S_0 \Rightarrow^* \square$

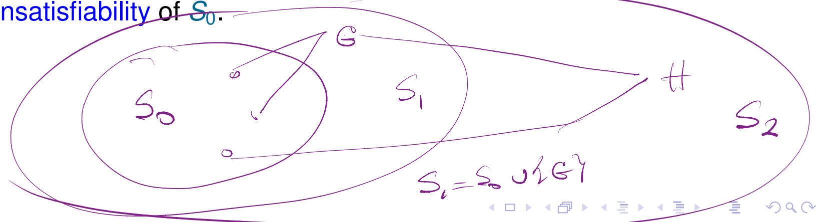
sound

Completeness is formulated in terms of derivability of the empty clause $\square$ from a set $S_0$ of clauses in an inference system $\mathbb{I}$. However, this formulations gives no hint on how to search for such a derivation.

Idea:

- Take a set of clauses $S$ (the search space), initially $S = S_0$. Repeatedly apply inferences in $\mathbb{I}$ to clauses in $S$ and add their conclusions to $S$, unless these conclusions are already in $S$.

- If, at any stage, we obtain $\square$, we terminate and report unsatisfiability of $S_0$.



# How to Establish Satisfiability?

$\square \equiv$ false

When can we report satisfiability?

# How to Establish Satisfiability?

When can we report satisfiability?

When we build a set $S$ such that any inference applied to clauses in $S$ is already a member of $S$. Any such set of clauses is called saturated (with respect to $\mathbb{I}$).

# How to Establish Satisfiability?

When can we report satisfiability?

When we build a set $S$ such that any inference applied to clauses in $S$ is already a member of $S$. Any such set of clauses is called saturated (with respect to $\mathbb{I}$).

In first-order logic it is often the case that all saturated sets are infinite (due to undecidability), so in practice we can never build a saturated set.

The process of trying to build one is referred to as saturation.

$f(a)=a$ $\quad S_0$ $\qquad\qquad f(a)\neq a$

$\Downarrow \qquad \Downarrow$

$ff(a)=fa \quad S_1$

$\Downarrow \qquad \Downarrow$

$fffa=ffa \quad S_2$

$\Downarrow$

$\vdots$

$f^n(a)=f^m(a), \quad \forall n,m \in \mathbb{N}$

$f(a) = a$ $\qquad$ $f^{100}(a) \neq a$

# Saturated Set of Clauses

Let $\mathbb{I}$ be an inference system on formulas and $S$ be a set of formulas.

- $S$ is called saturated with respect to $\mathbb{I}$, or simply $\mathbb{I}$-saturated, if for every inference of $\mathbb{I}$ with premises in $S$, the conclusion of this inference also belongs to $S$.

- The closure of $S$ with respect to $\mathbb{I}$, or simply $\mathbb{I}$-closure, is the smallest set $S'$ containing $S$ and saturated with respect to $\mathbb{I}$.

# Inference Process

Inference process: sequence of sets of formulas $S_0, S_1, \ldots$, denoted

$$S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \ldots$$

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

# Inference Process

Inference process: sequence of sets of formulas $S_0, S_1, \ldots$, denoted

$$S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \ldots$$

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

We say that this step is an $\mathbb{I}$-step if

1. there exists an inference

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

   in $\mathbb{I}$ such that $\{F_1, \ldots, F_n\} \subseteq S_i$;
2. $S_{i+1} = S_i \cup \{F\}$.

# Inference Process

Inference process: sequence of sets of formulas $S_0, S_1, \ldots$, denoted

$$S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \ldots$$

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

We say that this step is an $\mathbb{I}$-step if

1. there exists an inference

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

   in $\mathbb{I}$ such that $\{F_1, \ldots, F_n\} \subseteq S_i$;
2. $S_{i+1} = S_i \cup \{F\}$.

An $\mathbb{I}$-inference process is an inference process whose every step is an $\mathbb{I}$-step.

# Property

Let $S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \ldots$ 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 $S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \ldots$ is the set of formulas $\bigcup_i S_i$.

# Limit of a Process

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

In other words, the limit is the set of all derived formulas. We denote the limit by $S_\infty$.

# Limit of a Process

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

In other words, the limit is the set of all derived formulas. We denote the limit by $S_\infty$.

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 $S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \ldots$ is the set of formulas $\bigcup_i S_i$.

In other words, the limit is the set of all derived formulas. We denote the limit by $S_\infty$.

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?

No

# Fairness

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

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

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

# Limit of a Fair Inference Process

Let $S_0 \Rightarrow S_1 \Rightarrow S_2 \Rightarrow \ldots$ be an fair inference process using a sound inference system $\mathbb{I}$.

Exercise: Show that the limit of $S_\infty$ is the $\mathbb{I}$-closure of $S_0$.

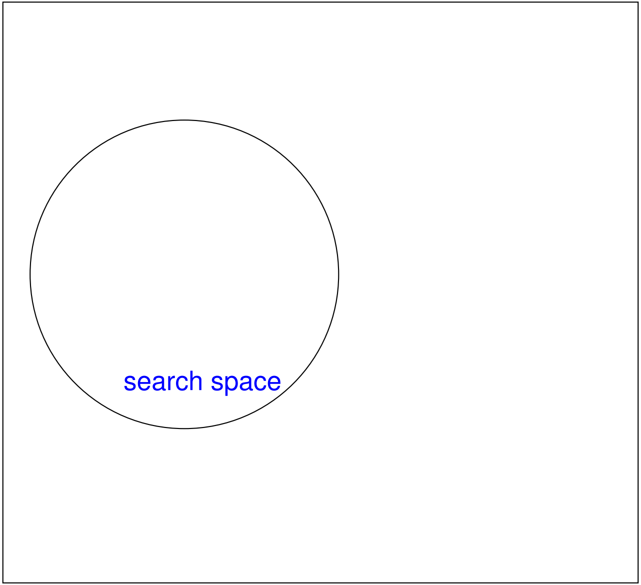
# Completeness, reformulated

**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 $S_\infty$ of this inference process contains $\square$.

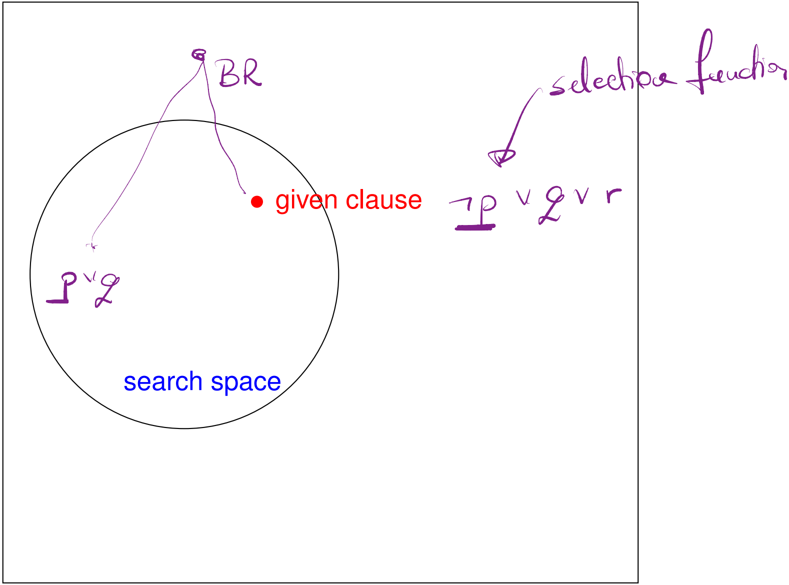
# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



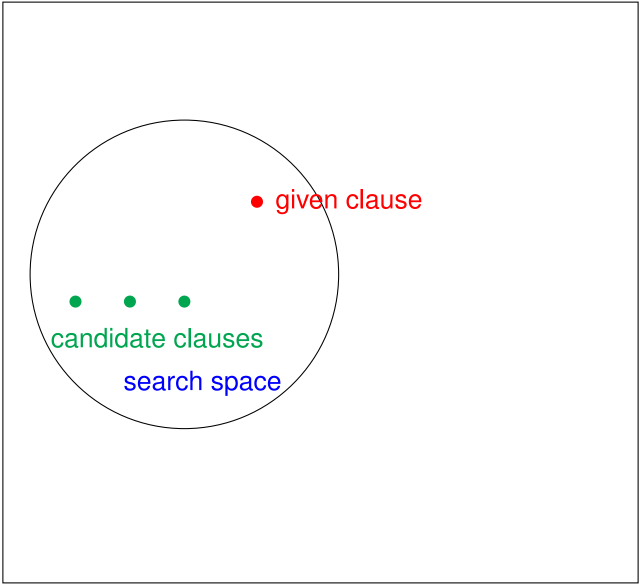
# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



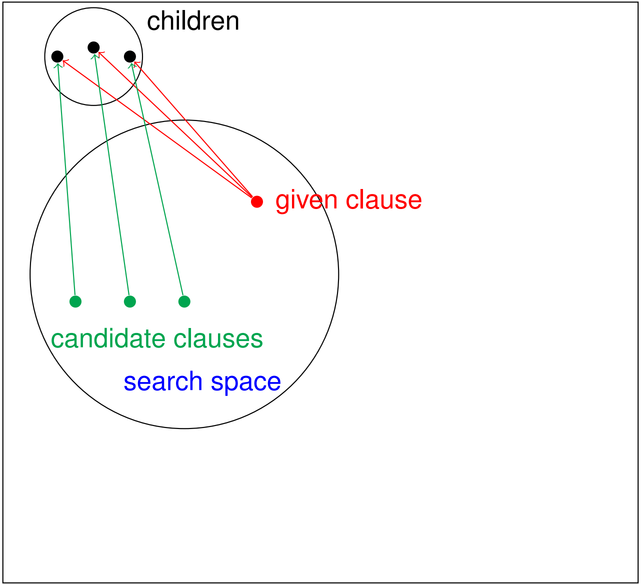
# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



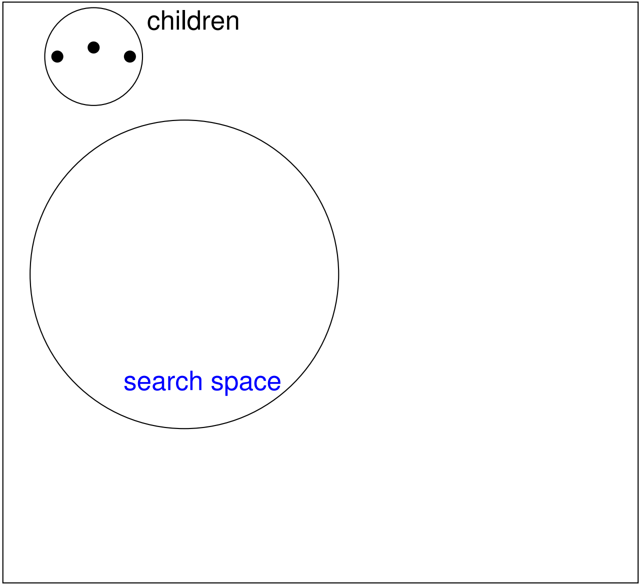
# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



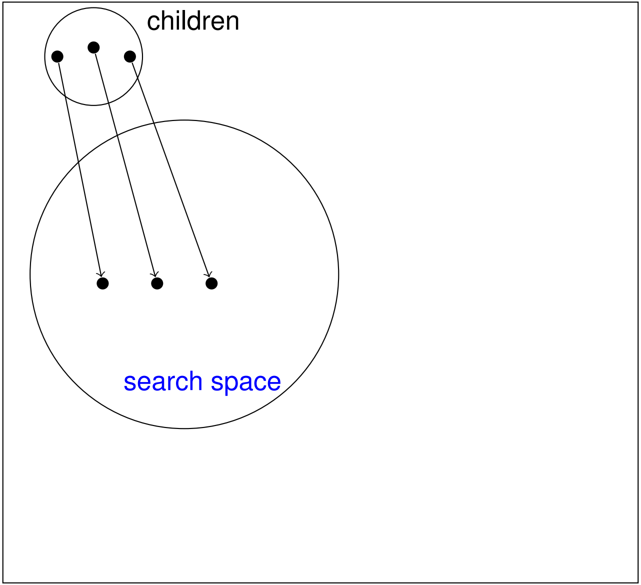
# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



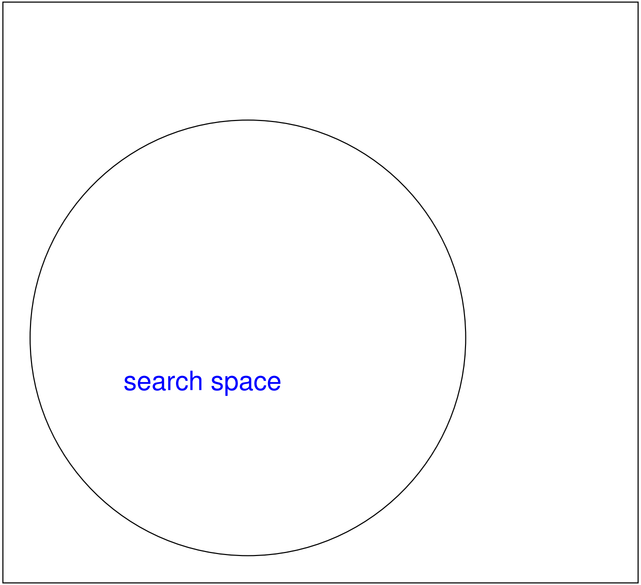
# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



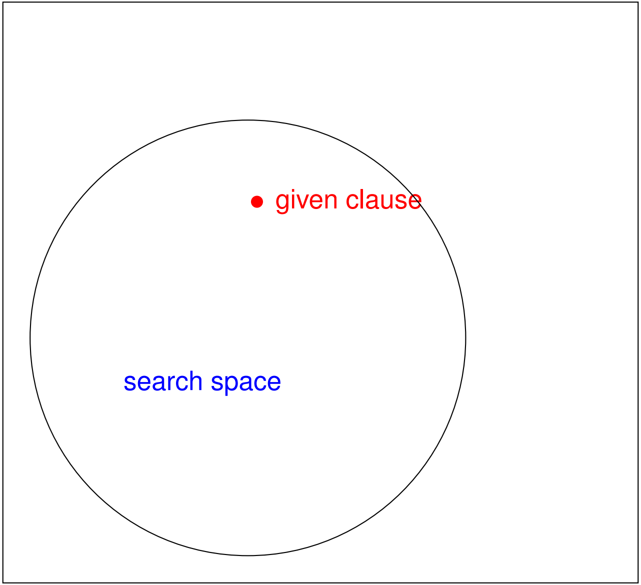
# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



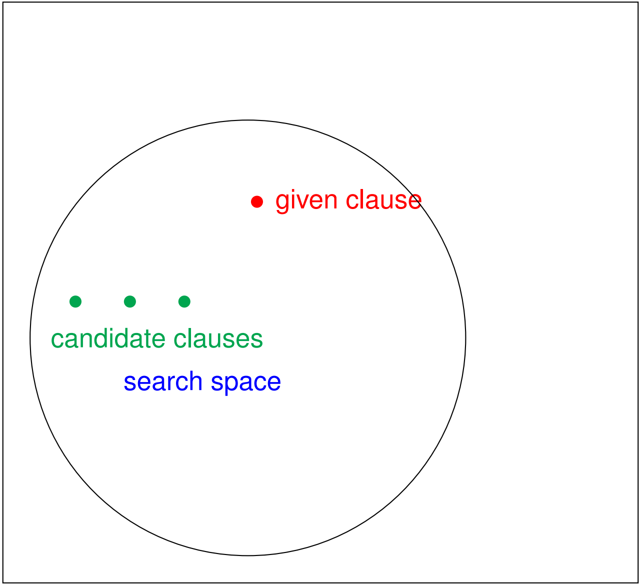
# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



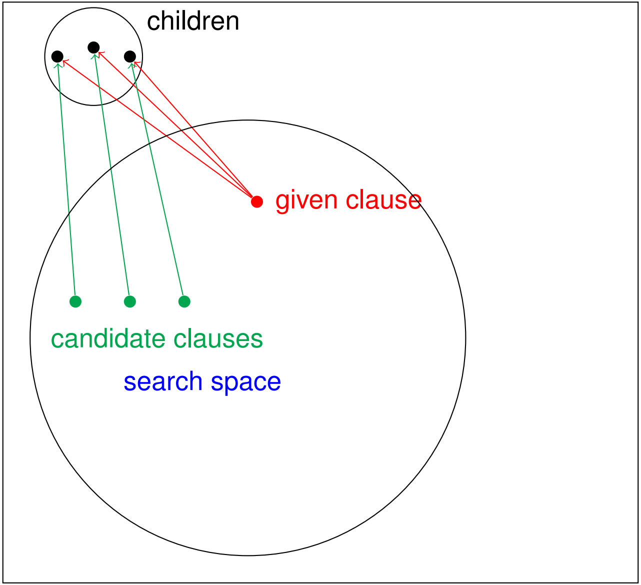
# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



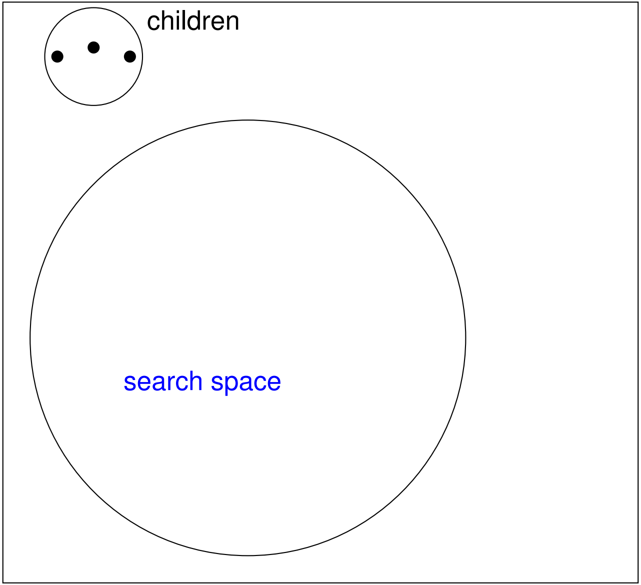
# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



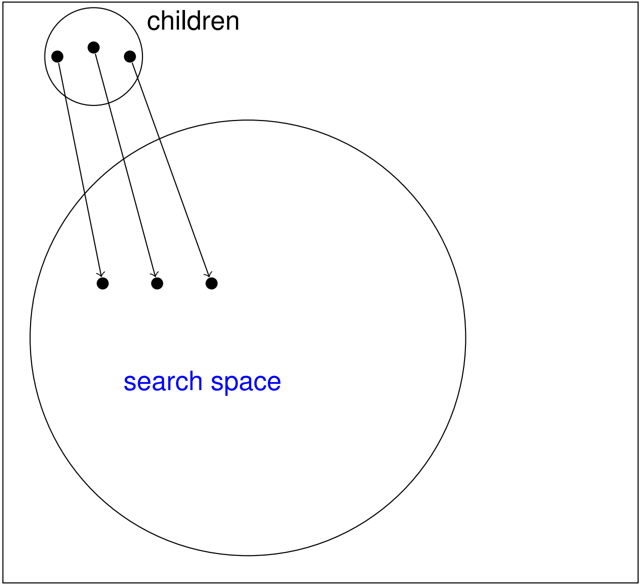
# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



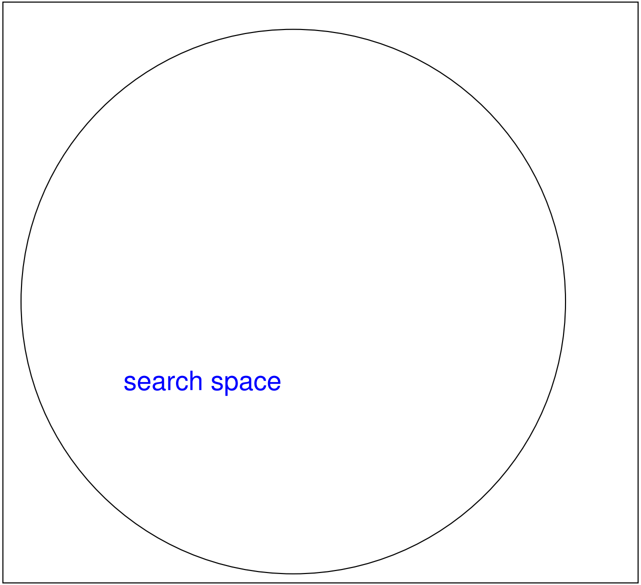
# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



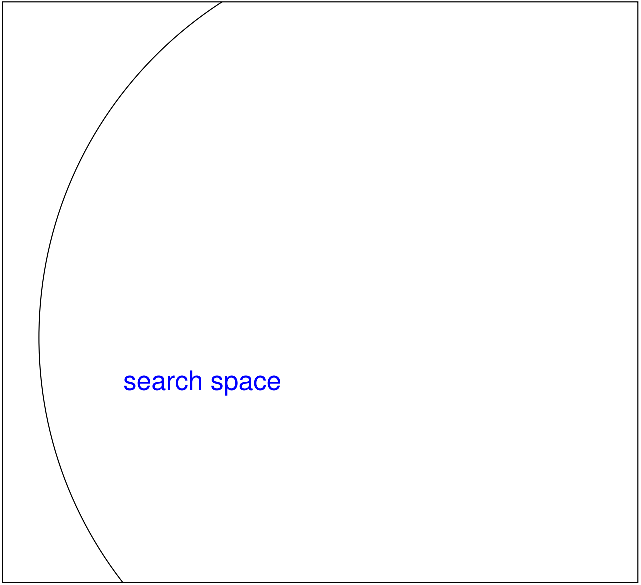
# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



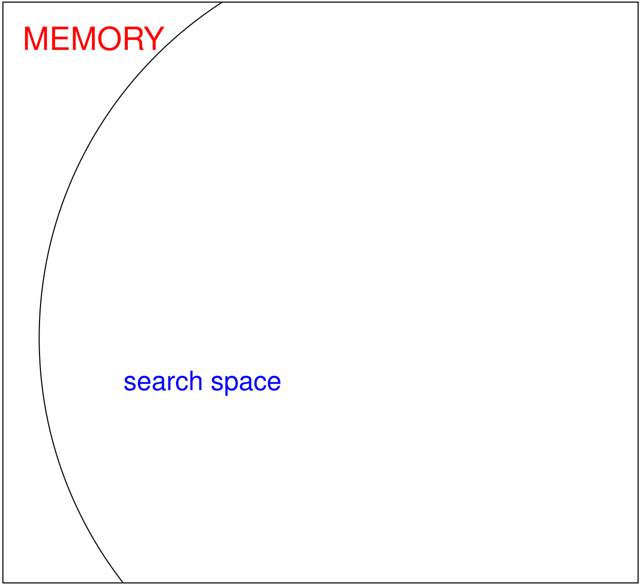
# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



# Fair Saturation Algorithms: Inference Selection by Clause Selection

(Given-Clause Algorithm)



# 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 $\Box$ is generated, in this case the input set of clauses is unsatisfiable.

2. Saturation will terminate without ever generating $\Box$, in this case the input set of clauses in satisfiable.

3. Saturation will run forever, but without generating $\Box$. 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 in 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.

$\neg p \supset p$

$p \vee \underline{\neg p}$

$\underline{p} \vee \underline{\neg p}$

$\cancel{\underline{p} \vee \neg p}$

# Saturation Algorithm

Even when we implement inference selection by clause selection, there are too many inferences, especially when the search space grows.

# Saturation Algorithm

Even when we implement inference selection by clause selection, there are too many inferences, especially when the search space grows.

Solution: only apply inferences to the selected clause and the previously selected clauses.

# Saturation Algorithm

Even when we implement inference selection by clause selection, there are too many inferences, especially when the search space grows.

Solution: only apply inferences to the selected clause and the previously selected clauses.

Thus, the search space is divided in two parts:

- active clauses, that participate in inferences;
- passive clauses, that do not participate in inferences.

Observation: the set of passive clauses is usually considerably larger than the set of active clauses, often by 2-4 orders of magnitude (depending on the saturation algorithm and the problem).

# Outline

Inference Systems with Selection (Recap)

Saturation Algorithms

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
$a \neq b \vee b \neq c \vee f(c, c) = f(a, a)$.

$\underbrace{a = b \wedge b = c}_{a = c} \rightarrow f(c,c) = f(a,a)$

# 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 $\cancel{C \vee D}$, 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.