Informatics

University of Southampton

Finding Connections via Satisfiability Solving

Clemens Eisenhofer (TU Wien)

Joint work with

Michael Rawson (U. Southampton) & *Laura Kovács* (TU Wien)

The Two Worlds

The Two Worlds

SAT	Most Deductive Calculi

The Two Worlds

SAT	Most Deductive Calculi
(Mostly) Efficient Reasoning	Construction (mostly) by enumeration/intuition
Very Easy Rules – CDCL (DPLL)	Can Be Very Complicated

The Two Worlds

SAT	Most Deductive Calculi
(Mostly) Efficient Reasoning	Construction (mostly) by enumeration/intuition
Very Easy Rules – CDCL (DPLL)	Can Be Very Complicated
E.g., $\vDash ((A \Rightarrow B) \Rightarrow (A \Rightarrow B))$?	

The Two Worlds

SAT	Most Deductive Calculi
(Mostly) Efficient Reasoning	Construction (mostly) by enumeration/intuition
Very Easy Rules – CDCL (DPLL)	Can Be Very Complicated
E.g., $\vDash ((A \Rightarrow B) \Rightarrow (A \Rightarrow B))$?	
CNF transformation: $\vdash_{CDCL} (\neg A \vee B) \wedge A \wedge \neg B$ 2x BCP: $\vdash_{CDCL} \bot$	

The Two Worlds

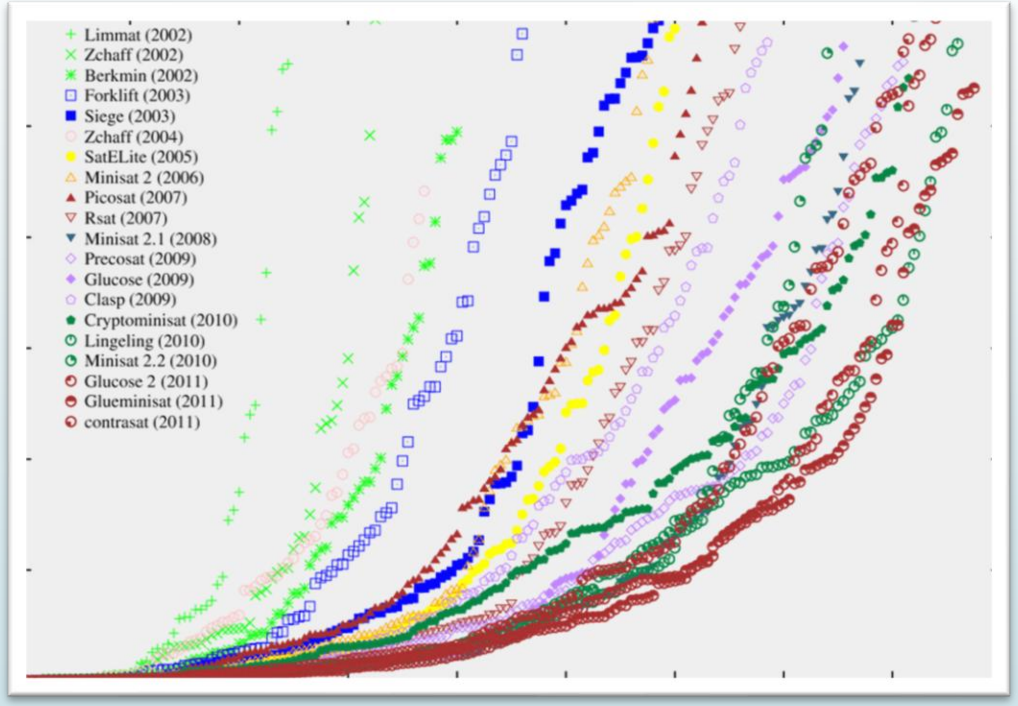
SAT	Most Deductive Calculi
(Mostly) Efficient Reasoning	Construction (mostly) by enumeration/intuition
Very Easy Rules – CDCL (DPLL)	Can Be Very Complicated
E.g., $\vDash ((\mathrm{A} \Rightarrow \mathrm{B}) \Rightarrow (\mathrm{A} \Rightarrow \mathrm{B}))$?	
CNF transformation: $\vdash_{CDCL} (\neg A \vee B) \wedge A \wedge \neg B$ 2x BCP: $\vdash_{CDCL} \perp$	$\vdash_{HC} (\mathrm{X} \Rightarrow ((\mathrm{X} \Rightarrow \mathrm{X}) \Rightarrow \mathrm{X}) \Rightarrow ((\mathrm{X} \Rightarrow (\mathrm{X} \Rightarrow \mathrm{X})) \Rightarrow (\mathrm{X} \Rightarrow \mathrm{X}))$ $\vdash_{HC} X \Rightarrow ((X \Rightarrow X) \Rightarrow X)$ $\vdash_{HC} (X \Rightarrow (X \Rightarrow X)) \Rightarrow (X \Rightarrow X)$ … $\vdash_{HC} \mathrm{X} \Rightarrow \mathrm{X}$ and use $\mathrm{X} \coloneqq \mathrm{A} \Rightarrow \mathrm{B}$

The Two Worlds

Theory of Proof Objects

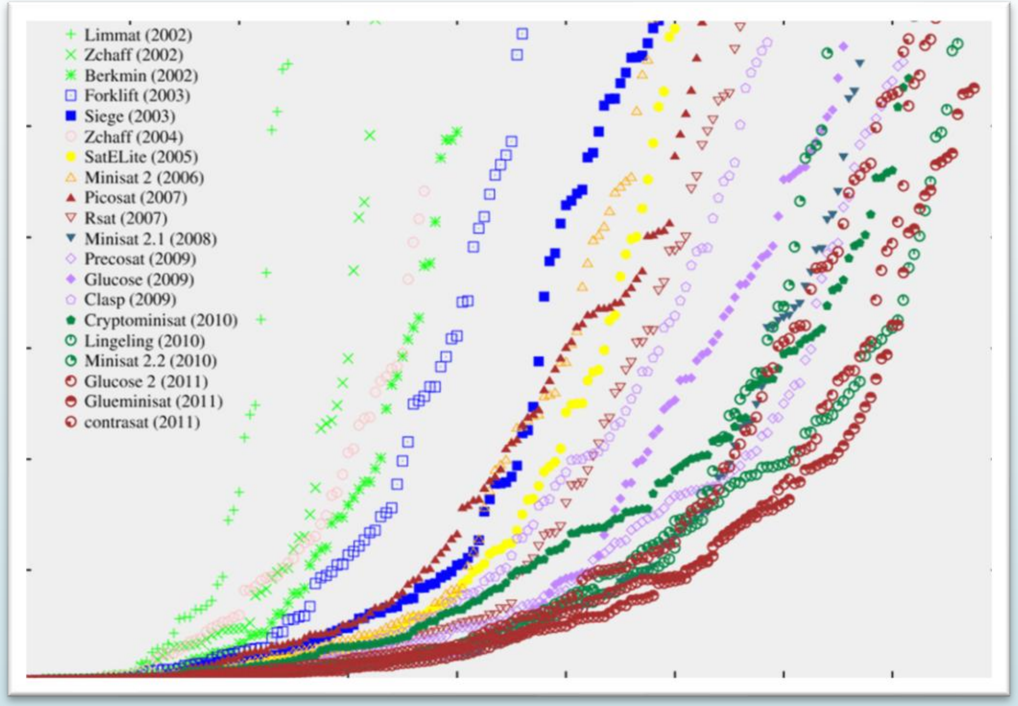
One Step back: Why Combine them?

SAT Solvers are efficient



One Step back: Why Combine them?

SAT Solvers are efficient



… and get more efficient every year

Get to the Point!

Get to the Point!

- We suggested using SAT-Solvers for doing **proof enumeration** for **connection calculus**:

  Embedding the Connection Calculus in Satisfiability Modulo Theories [AReCCa@TABLEAUX'23]

Get to the Point!

- We suggested using SAT-Solvers for doing **proof enumeration** for **connection calculus**:

  Embedding the Connection Calculus in Satisfiability Modulo Theories [AReCCa@TABLEAUX’23]

- We **continued** on that end and here is what came out…

First-Order Connection Calculus (CC)

First-Order Connection Calculus (CC)

- "First calculus towards ATP" [Bibel; Loveland – late 70s]

First-Order Connection Calculus (CC)

- "First calculus towards ATP" [Bibel; Loveland – late 70s]
- Variant of (**first-order) tableaux** and **binary resolution**

First-Order Connection Calculus (CC)

- "First calculus towards ATP" [Bibel; Loveland – late 70s]
- Variant of (**first-order) tableaux** and **binary resolution**
- **Sound** & **complete**

First-Order Connection Calculus (CC)

- "First calculus towards ATP" [Bibel; Loveland – late 70s]
- Variant of (**first-order) tableaux** and **binary resolution**
- **Sound** & **complete**
- Unlike "ordinary" tableaux or resolution: **Goal-directed & Non-confluent**
  - Some proof steps are wrong and result in "dead ends"

First-Order Connection Calculus (CC)

- "First calculus towards ATP" [Bibel; Loveland – late 70s]
- Variant of (**first-order) tableaux** and **binary resolution**
- **Sound** & **complete**
- Unlike "ordinary" tableaux or resolution: **Goal-directed & Non-confluent**
  - Some proof steps are wrong and result in "dead ends"
    - E.g.

$$\boldsymbol{A}, \quad (\neg A \lor B), \quad (\neg A \lor C), \quad \neg C$$

First-Order Connection Calculus (CC)

- "First calculus towards ATP" [Bibel; Loveland – late 70s]
- Variant of (**first-order) tableaux** and **binary resolution**
- **Sound** & **complete**
- Unlike "ordinary" tableaux or resolution: **Goal-directed & Non-confluent**
  - Some proof steps are wrong and result in "dead ends"
    - E.g.

$$\boldsymbol{A}, \quad (\neg A \lor B), \quad (\neg A \lor C), \quad \neg C$$

    - Resolve A with
    - $(\neg A \lor B)$ resulting in B **or**
    - $(\neg A \lor C)$ resulting in C

First-Order Connection Calculus (CC)

- "First calculus towards ATP" [Bibel; Loveland – late 70s]
- Variant of (**first-order) tableaux** and **binary resolution**
- **Sound** & **complete**
- Unlike "ordinary" tableaux or resolution: **Goal-directed & Non-confluent**
  - Some proof steps are wrong and result in "dead ends"
    - E.g.

      $$\boldsymbol{A}, \quad (\neg \mathrm{A} \vee \mathrm{B}), \quad (\neg \mathrm{A} \vee \mathrm{C}), \quad \neg \mathrm{C}$$

    - Resolve A with
    - $(\neg A \vee B)$ resulting in B **or**
    - $(\neg A \vee \mathrm{C})$ resulting in C
    - However, we cannot do anything with B (dead end)

First-Order Connection Calculus (CC)

- "First calculus towards ATP" [Bibel; Loveland – late 70s]
- Variant of (**first-order) tableaux** and **binary resolution**
- **Sound** & **complete**
- Unlike "ordinary" tableaux or resolution: **Goal-directed & Non-confluent**
  - Some proof steps are wrong and result in "dead ends"
    - E.g.

      $$\boldsymbol{A}, \quad (\neg A \lor B), \quad (\neg A \lor C), \quad \neg C$$

    - Resolve A with
    - $(\neg A \lor B)$ resulting in B **or**
    - $(\neg A \lor C)$ resulting in C
    - However, we cannot do anything with B (dead end)
- Iterative deepening & proof enumeration

The Matrix Representation

The Matrix Representation

- Alternative way of presenting CC proofs

The Matrix Representation

- Alternative way of presenting CC proofs
- Clause copies are "written" vertically

The Matrix Representation

- Alternative way of presenting CC proofs
- Clause copies are "written" vertically
- Lines are drawn between dual literals
  - The literals are "connected"

The Matrix Representation

- Alternative way of presenting CC proofs
- Clause copies are "written" vertically
- Lines are drawn between dual literals
  - The literals are "connected"

C_1: $\forall x \forall y \left(\neg P(x) \vee \neg P\big(f(x)\big)\right)$

C_2: $\forall z \left(P(z) \vee P\big(f(z)\big)\right)$

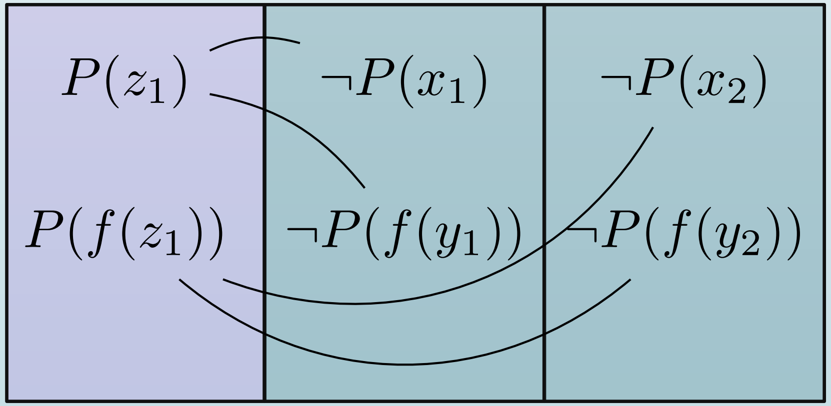
$P(z_1)$	$\neg P(x_1)$	$\neg P(x_2)$
$P(f(z_1))$	$\neg P(f(y_1))$	$\neg P(f(y_2))$

The Matrix Representation

- Alternative way of presenting CC proofs
- Clause copies are "written" vertically
- Lines are drawn between dual literals
  - The literals are "connected"

C_1: $\forall x \forall y \left(\neg P(x) \lor \neg P(f(x))\right)$

C_2: $\forall z \left(P(z) \lor P(f(z))\right)$



$\sigma(z_1) \mapsto f(y_1)$, $\sigma(x_1) \mapsto f(y_1)$,
$\sigma(x_2) \mapsto f(f(y_1))$, $\sigma(y_2) \mapsto f(y_1)$

The Matrix Representation

The Matrix Representation

- Set of connections – “the matrix" – is **spanning**: **no open path**

The Matrix Representation

- Set of connections – “the matrix" – is **spanning**: **no open path**
- **Open path:** One **literal of each clause** instance,

  such that **none of them is connected** with each other

The Matrix Representation

- Set of connections – “the matrix" – is **spanning**: **no open path**
- **Open path:** One **literal of each clause** instance,

  such that **none of them is connected** with each other

$$Open(P) \equiv \big(\forall C \exists L (L \in P)\big) \land \forall L_1, L_2 (L_1 \nsim L_2)$$

The Matrix Representation

- Set of connections – “the matrix" – is **spanning**: **no open path**
- **Open path:** One **literal of each clause** instance,

  such that **none of them is connected** with each other

$$Open(P) \equiv \big(\forall C \exists L (L \in P)\big) \land \forall L_1, L_2 (L_1 \nsim L_2)$$

- **Only spanning matrices** are **correct** proofs!

The Matrix Representation

- Set of connections – “the matrix" – is **spanning**: **no open path**
- **Open path:** One **literal of each clause** instance,

  such that **none of them is connected** with each other

$$Open(P) \equiv \big(\forall C \exists L (L \in P)\big) \land \forall L_1, L_2 (L_1 \nsim L_2)$$

- **Only spanning matrices** are **correct** proofs!

- Set of connections – "the matrix" – is **fully connected:**

The Matrix Representation

- Set of connections – “the matrix" – is **spanning**: **no open path**
- **Open path:** One **literal of each clause** instance,

  such that **none of them is connected** with each other

$$Open(P) \equiv \big(\forall C \exists L (L \in P)\big) \land \forall L_1, L_2 (L_1 \nsim L_2)$$

- **Only spanning matrices** are **correct** proofs!

- Set of connections – "the matrix" – is **fully connected:**

  each literal is connected with at least one other literal

Another Example

Another Example

$$P(z_1) \quad \neg P(x_1) \quad \boldsymbol{\neg P(x_2)}$$

$$\boldsymbol{P(f(z_1))} \quad \boldsymbol{\neg P(f(y_1))} \quad \neg P(f(y_2))$$

Another Example

$$P(z_1) \quad \neg P(x_1) \quad \boldsymbol{\neg P(x_2)}$$

$$\boldsymbol{P(f(z_1))} \quad \boldsymbol{\neg P(f(y_1))} \quad \neg P(f(y_2))$$

$$P(z_1) \quad \neg P(x_1) \quad \neg P(x_2)$$

$$P(f(z_1)) \quad \neg P(f(y_1)) \quad \neg P(f(y_2))$$

The Matrix Representation - Definitions

C_1: $\forall x \forall y \left(\neg P(x) \vee \neg P(f(x))\right)$

C_2: $\forall z \left(P(z) \vee P(f(z))\right)$

$$\begin{matrix} P(z_1) & \neg P(x_1) & \neg P(x_2) \\ P(f(z_1)) & \neg P(f(y_1)) & \neg P(f(y_2)) \end{matrix}$$

$\sigma(z_1) \mapsto f(y_1)$, $\sigma(x_1) \mapsto f(y_1)$,
$\sigma(x_2) \mapsto f(f(y_1))$, $\sigma(y_2) \mapsto f(y_1)$

The Matrix Representation - Definitions

Clauses

$$C_1:\ \forall x \forall y \left(\neg P(x) \vee \neg P\big(f(x)\big)\right)$$

$$C_2: \forall z \left(P(z) \vee P\big(f(z)\big)\right)$$

$$\begin{matrix} P(z_1) & \neg P(x_1) & \neg P(x_2) \\ P(f(z_1)) & \neg P(f(y_1)) & \neg P(f(y_2)) \end{matrix}$$

$$\sigma(z_1) \mapsto f(y_1), \quad \sigma(x_1) \mapsto f(y_1),$$
$$\sigma(x_2) \mapsto f\big(f(y_1)\big), \quad \sigma(y_2) \mapsto f(y_1)$$

The Matrix Representation - Definitions

C_1: $\forall x \forall y \left(\neg P(x) \vee \neg P(f(x))\right)$

C_2: $\forall z \left(P(z) \vee P(f(z))\right)$

$$\begin{array}{ccc} P(z_1) & \neg P(x_1) & \neg P(x_2) \\ P(f(z_1)) & \neg P(f(y_1)) & \neg P(f(y_2)) \end{array}$$

Unifier

$$\sigma(z_1) \mapsto f(y_1), \quad \sigma(x_1) \mapsto f(y_1),$$
$$\sigma(x_2) \mapsto f(f(y_1)), \quad \sigma(y_2) \mapsto f(y_1)$$

The Matrix Representation - Definitions

Clause Instances

$$\begin{array}{ccc} P(z_1) & \neg P(x_1) & \neg P(x_2) \\ P(f(z_1)) & \neg P(f(y_1)) & \neg P(f(y_2)) \end{array}$$

$$C_1 : \forall x \forall y \left(\neg P(x) \vee \neg P(f(x)) \right)$$

$$C_2 : \forall z \left(P(z) \vee P(f(z)) \right)$$

$\sigma(z_1) \mapsto f(y_1)$, $\sigma(x_1) \mapsto f(y_1)$,

$\sigma(x_2) \mapsto f(f(y_1))$, $\sigma(y_2) \mapsto f(y_1)$

The Matrix Representation - Definitions

C_1: $\forall x \forall y \left(\neg P(x) \vee \neg P(f(x))\right)$

C_2: $\forall z \left(P(z) \vee P(f(z))\right)$

$$
\begin{matrix}
P(z_1) & \neg P(x_1) & \neg P(x_2) \\
P(f(z_1)) & \neg P(f(y_1)) & \neg P(f(y_2))
\end{matrix}
$$

$\sigma(z_1) \mapsto f(y_1)$, $\sigma(x_1) \mapsto f(y_1)$,

$\sigma(x_2) \mapsto f(f(y_1))$, $\sigma(y_2) \mapsto f(y_1)$

Connections

The Matrix Representation - Definitions

C_1: $\forall x \forall y \left(\neg P(x) \lor \neg P(f(x))\right)$

C_2: $\forall z \left(P(z) \lor P(f(z))\right)$

$$\begin{array}{ccc} P(z_1) & \neg P(x_1) & \neg P(x_2) \\ P(f(z_1)) & \neg P(f(y_1)) & \neg P(f(y_2)) \end{array}$$

$\sigma(z_1) \mapsto f(y_1)$, $\sigma(x_1) \mapsto f(y_1)$,
$\sigma(x_2) \mapsto f(f(y_1))$, $\sigma(y_2) \mapsto f(y_1)$

Depth/Size = 3

The Matrix Representation - Definitions

C_1: $\forall x \forall y \left(\neg P(x) \vee \neg P(f(x))\right)$

C_2: $\forall z \left(P(z) \vee P(f(z))\right)$

$$\begin{array}{ccc} P(z_1) & \neg P(x_1) & \neg P(x_2) \\ P(f(z_1)) & \neg P(f(y_1)) & \neg P(f(y_2)) \end{array}$$

$\sigma(z_1) \mapsto f(y_1)$, $\sigma(x_1) \mapsto f(y_1)$,
$\sigma(x_2) \mapsto f(f(y_1))$, $\sigma(y_2) \mapsto f(y_1)$

Depth/Size = 3

SAT/SMT Encodings for CC

Finding Connections via Satisfiability Solving

The CC Encodings

The CC Encodings

- In the paper we discussed three different encodings

The CC Encodings

- In the paper we discussed three different encodings

1. CC tableau trees (omitted for *this* talk)
2. **CC matrices with static sizes**
3. **CC matrices with dynamic sizes**

Basic Matrix Encoding Idea

Basic Matrix Encoding Idea

We **want to encode** for some fixed number $d > 0$:

1. The matrix contains **exactly *d* clause instances**
2. The matrix is spanning

Basic Matrix Encoding Idea

We **want to encode** for some fixed number $d > 0$:

1. The matrix contains **exactly *d* clause instances**
2. The matrix is spanning

Condition 2. cannot be really expressed directly, hence we use instead:

1. The matrix contains **exactly *d* clause instances**
2. The matrix is **fully connected**

Basic Matrix Encoding Idea

We **want to encode** for some fixed number *d > 0*:

1. The matrix contains **exactly *d* clause instances**
2. The matrix is spanning

Condition 2. cannot be really expressed directly, hence we use instead:

1. The matrix contains **exactly *d* clause instances**
2. The matrix is **fully connected**

„**Fully connected**“ is used as an **approximation** for „**spanning**“.

Basic Matrix Encoding Idea

We **want to encode** for some fixed number *d > 0*:

1. The matrix contains **exactly *d* clause instances**
2. The matrix is spanning

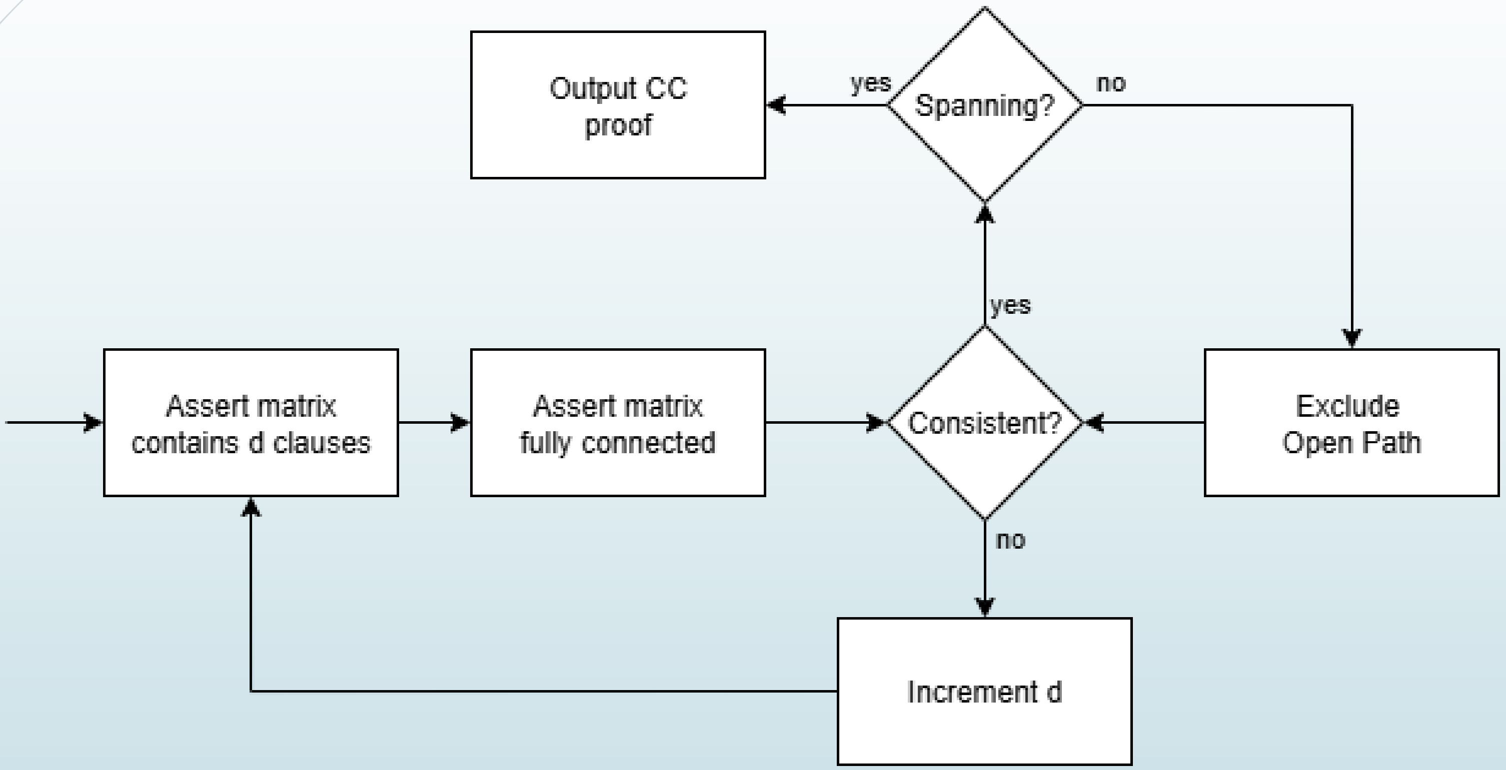
Condition 2. cannot be really expressed directly, hence we use instead:

1. The matrix contains **exactly *d* clause instances**
2. The matrix is **fully connected**

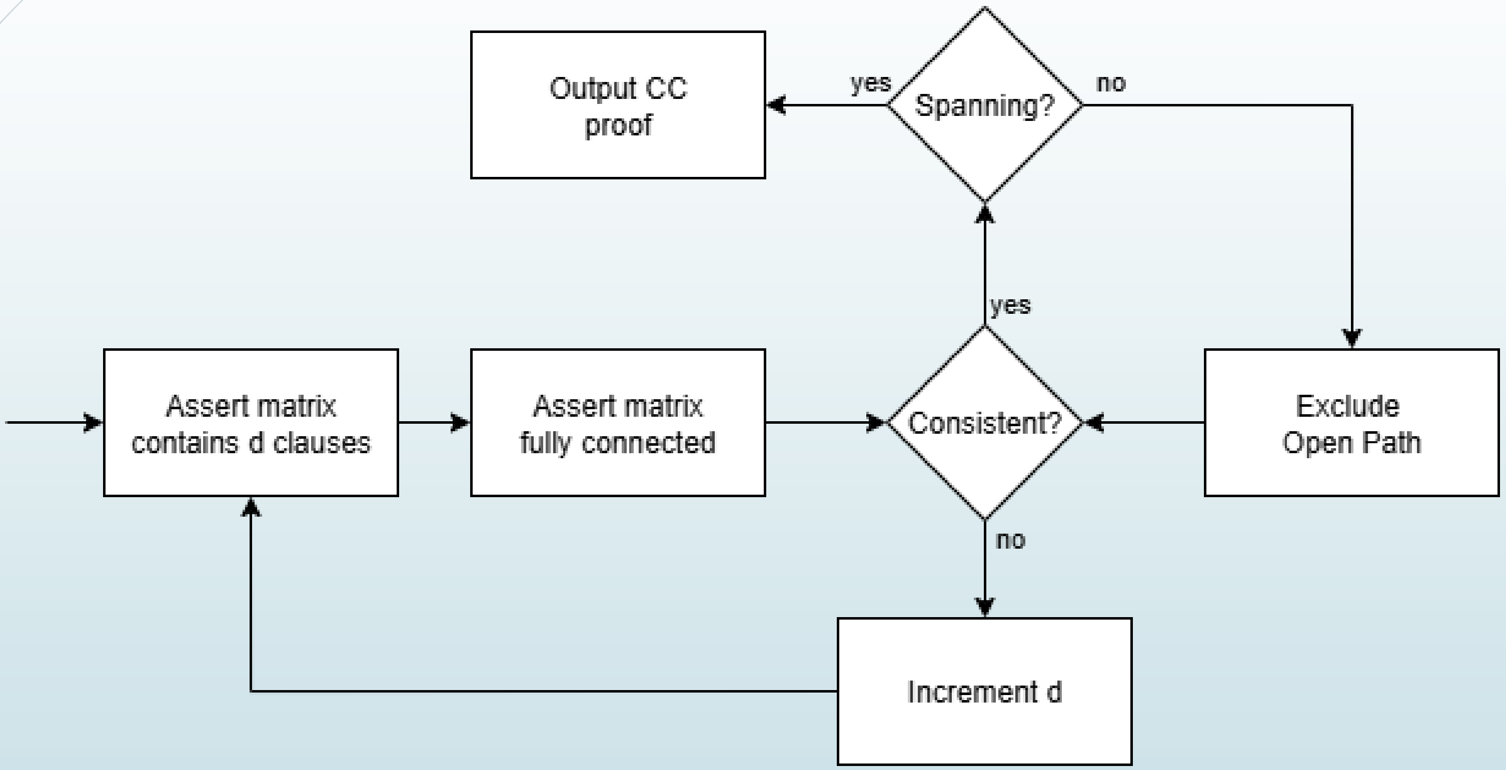
„**Fully connected**“ is used as an **approximation** for „**spanning**“.

➔ We check for spanningness and rule out spurious proofs on-demand

Basic Matrix Encoding Idea



Basic Matrix Encoding Idea



- **Encoding is lazy** (SMT style)
- Unification can be expressed using **algebraic datatypes** (ADT)

More Precisely

More Precisely

- We have two kinds of literals:

More Precisely

- We have two kinds of literals:
  1. Selectors
     - e.g., $S^1_{P(z) \vee P(f(z))}$ denotes $P(z_1) \vee P(f(z_1))$ occurs in the matrix

More Precisely

- We have two kinds of literals:
  1. Selectors
     - e.g., $S^1_{P(z)\vee P\left(f(z)\right)}$ denotes $P(z_1) \vee P\left(f(z_1)\right)$ occurs in the matrix
  2. Connectors
     - E.g., $\langle P(f(z_1)) \sim \neg P(x_1) \rangle$ denotes that the given literal instances are dual ("connected")

More Precisely

- We have two kinds of literals:
  1. Selectors
     - e.g., $S^1_{P(z)\vee P(f(z))}$ denotes $P(z_1) \vee P\big(f(z_1)\big)$ occurs in the matrix
  2. Connectors
     - E.g., $\langle P(f(z_1)) \sim \neg P(x_1)\rangle$ denotes that the given literal instances are dual ("connected")
     - i.e., enforces $f(z_1)$ unifies with x_1

The Whole Encoding

The Whole Encoding

1. There are *d* clause instance in the matrix:

$$|\{S_C^i \mid C, 1 \leq i \leq d\}| = d$$

The Whole Encoding

1. There are *d* clause instance in the matrix:

$$|\{S_C^i \mid C, 1 \leq i \leq d\}| = d$$

2. For each literal S^i in the matrix there is a connected one (fully connected)

$$S^i \Rightarrow \bigvee_{D} \bigvee_{1 \leq k \leq d} \bigvee_{K \in D^k} \left(S_D^k \wedge \langle L \sim K \rangle\right)$$

The Whole Encoding

1. There are *d* clause instance in the matrix:

$$|\{S_C^i \mid C, 1 \leq i \leq d\}| = d$$

2. For each literal S^i in the matrix there is a connected one (fully connected)

$$S^i \Rightarrow \bigvee_{D} \bigvee_{1 \leq k \leq d} \bigvee_{K \in D^k} \left(S_D^k \wedge \langle L \sim K \rangle\right)$$

3. For every open path U and set of *d* selectors $\bar{S}$ we add

$$\bigwedge_{S \in \bar{S}} S \Rightarrow \bigvee_{\{L,K\} \subseteq U} \langle L \sim K \rangle$$

The Encoding

The Encoding

- **Sound**

The Encoding

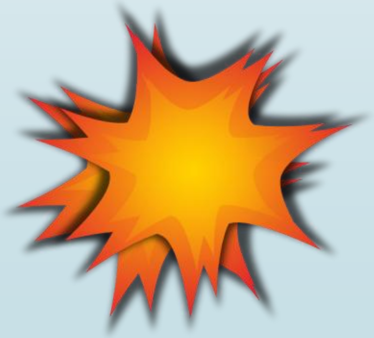
- **Sound**
- For some fixed *d*
  - **Terminating**
  - Complexity Σ_2^P-complete [„NP given we have a co-NP oracle“]

The Encoding

- **Sound**
- For some fixed *d*
  - **Terminating**
  - Complexity Σ_2^P-complete [„NP given we have a co-NP oracle“]
- **Completeness** by stepwise incrementing *d*

The Encoding

- **Sound**
- For some fixed *d*
  - **Terminating**
  - Complexity Σ_2^P-complete [„NP given we have a co-NP oracle“]
- **Completeness** by stepwise incrementing *d*
- However, incrementing *d* by one ➔ one more copy of *each clause*
  - Makes it even more explosive



Dynamic Matrix Sizing Idea

Dynamic Matrix Sizing Idea

- One **counter $\boldsymbol{\mu(C)}$ for each clause** C
  - So far, we had for all C that $\mu(C) \coloneqq d$
  - Thus, we have S_C^i for any $1 \leq i \leq \mu(C)$
  - Literals $S_C^{\mu(\mathrm{C})+1}$ are **assumed** to be false

Dynamic Matrix Sizing Idea

- One **counter $\boldsymbol{\mu(C)}$ for each clause C**
  - So far, we had for all C that $\mu(C) \coloneqq d$
  - Thus, we have S_C^i for any $1 \le i \le \mu(C)$
  - Literals $S_C^{\mu(\mathrm{C})+1}$ are **assumed** to be false
- We **increment only some** $\mu(C)$ on failed proof attempts

Dynamic Matrix Sizing Idea

- One **counter** $\boldsymbol{\mu(C)}$ **for each clause** C
  - So far, we had for all C that $\mu(C) \coloneqq d$
  - Thus, we have S_C^i for any $1 \leq i \leq \mu(C)$
  - Literals $S_C^{\mu(\mathrm{C})+1}$ are **assumed** to be false
- We **increment only some** $\mu(C)$ on failed proof attempts
- Decisions are based on conflict analysis (**unsat core**)

$$S^i \Rightarrow \bigvee_{D} \bigvee_{1 \leq k \leq \boldsymbol{\mu(D)}+1} \bigvee_{K \in D^k} \left(S_D^k \wedge \langle L \sim K \rangle\right)$$

Dynamic Matrix Sizing Idea

- One **counter $\boldsymbol{\mu(C)}$ for each clause** C
  - So far, we had for all C that $\mu(C) \coloneqq d$
  - Thus, we have S_C^i for any $1 \leq i \leq \mu(C)$
  - Literals $S_C^{\mu(\mathrm{C})+1}$ are **assumed** to be false
- We **increment only some** $\mu(C)$ on failed proof attempts
- Decisions are based on conflict analysis (**unsat core**)

$$S^i \Rightarrow \bigvee_{D} \bigvee_{1 \leq k \leq \boldsymbol{\mu}(\boldsymbol{D})+1} \bigvee_{K \in D^k} \left(S_D^k \wedge \langle L \sim K \rangle\right)$$

- „Connect to some literal instance of clause D or require more instances of D“

Incrementing Capacity

Incrementing Capacity

- Consider the clause set

$$\{\, P(a), \qquad \neg P(x) \vee P\big(f(x)\big), \qquad \neg P(a) \,\}$$

$$\mu\big(P(a)\big) \mapsto 1$$
$$\mu\Big(P(x) \vee \neg P\big(f(x)\big)\Big) \mapsto 0$$
$$\mu\big(\neg P(a)\big) \mapsto 0$$

Incrementing Capacity

- Consider the clause set

$$\{\, P(a), \quad \neg P(x) \vee P\big(f(x)\big), \quad \neg P(a) \,\}$$

$P(a)$

$$\mu\big(P(a)\big) \mapsto 1$$
$$\mu\Big(P(x) \vee \neg P\big(f(x)\big)\Big) \mapsto 0$$
$$\mu\big(\neg P(a)\big) \mapsto 0$$

$$S^1{}_{P(a)} \Rightarrow \Big(S^1_{\neg P(x) \vee P(f(x))} \wedge \langle P(a) \sim \neg P(x_1)\rangle\Big) \vee S^1_{\neg P(a)}$$

Incrementing Capacity

- Consider the clause set

$$\{\, P(a), \qquad \neg P(x) \vee P\big(f(x)\big), \qquad \neg P(a) \,\}$$

$P(a)$

$$\mu\big(P(a)\big) \mapsto 1$$
$$\mu\Big(P(x) \vee \neg P\big(f(x)\big)\Big) \mapsto 0$$
$$\mu\big(\neg P(a)\big) \mapsto 0$$
$$\{\, S^1_{\neg P(x) \vee P(f(x))}, \qquad S^1_{\neg P(a)} \,\}$$

$$S^1{}_{P(a)} \Rightarrow \Big(S^1_{\neg P(x) \vee P(f(x))} \wedge \langle P(a) \sim \neg P(x_1)\rangle\Big) \vee S^1_{\neg P(a)}$$

Incrementing Capacity

- Consider the clause set

$$\{\, P(a), \quad \neg P(x) \vee P\big(f(x)\big), \quad \neg P(a) \,\}$$

$$P(a) \text{ —— } \neg P(x_1)$$

$$P(f(x_1))$$

$$\mu\big(P(a)\big) \mapsto 1$$
$$\mu\Big(P(x) \vee \neg P\big(f(x)\big)\Big) \mapsto 1$$
$$\mu\big(\neg P(a)\big) \mapsto 0$$

$$S^1{}_{P(a)} \Rightarrow \Big(S^1_{\neg P(x) \vee P(f(x))} \wedge \langle P(a) \sim \neg P(x_1)\rangle\Big) \vee S^1_{\neg P(a)}$$
$$S^1_{\neg P(x) \vee P(f(x))} \Rightarrow \Big(S^2_{\neg P(x) \vee P(f(x))} \wedge \big\langle P\big(f(x_1)\big) \sim P(x_2)\big\rangle\Big)$$

Incrementing Capacity

- Consider the clause set

$$\{\, P(a), \quad \neg P(x) \vee P\big(f(x)\big), \quad \neg P(a) \,\}$$

$$P(a) \text{ —— } \neg P(x_1)$$

$$P(f(x_1))$$

$$\mu\big(P(a)\big) \mapsto 1$$
$$\mu\Big(P(x) \vee \neg P\big(f(x)\big)\Big) \mapsto 1$$
$$\mu\big(\neg P(a)\big) \mapsto 0$$
$$\{\, S^2_{\neg P(x) \vee P(f(x))}, \quad S^1_{\neg P(a)} \,\}$$

$$S^1{}_{P(a)} \Rightarrow \Big(S^1_{\neg P(x) \vee P(f(x))} \wedge \langle P(a) \sim \neg P(x_1) \rangle\Big) \vee S^1_{\neg P(a)}$$
$$S^1_{\neg P(x) \vee P(f(x))} \Rightarrow \Big(S^2_{\neg P(x) \vee P(f(x))} \wedge \big\langle P\big(f(x_1)\big) \sim P(x_2) \big\rangle\Big)$$

Incrementing Capacity

- Consider the clause set

$$\{\, P(a), \qquad \neg P(x) \lor P\big(f(x)\big), \qquad \neg P(a) \,\}$$

$$P(a) \;—\; \neg P(x_1) \qquad \neg P(x_2)$$

$$P(f(x_1)) \qquad P(f(x_2))$$

$$\mu\big(P(a)\big) \mapsto 1$$

$$\mu\Big(P(x) \lor \neg P\big(f(x)\big)\Big) \mapsto 2$$

$$\mu\big(\neg P(a)\big) \mapsto 0$$

Incrementing Capacity

➔ Fair incrementation of clauses in unsat cores

- Consider the clause set

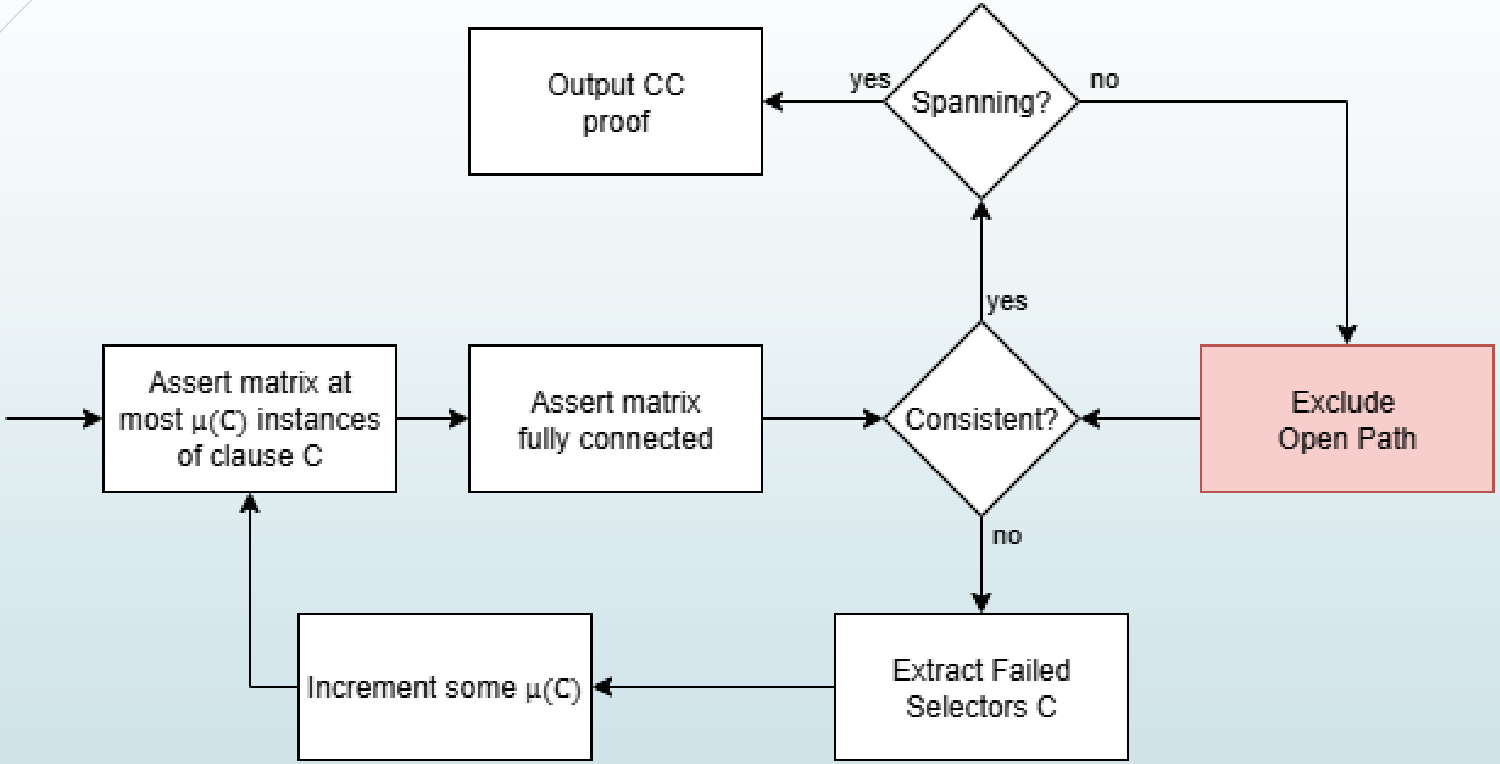
$$\{\, P(a), \qquad \neg P(x) \vee P\big(f(x)\big), \qquad \neg P(a) \,\}$$

$P(a)$ —— $\neg P(x_1)$ $\qquad$ $\neg P(x_2)$

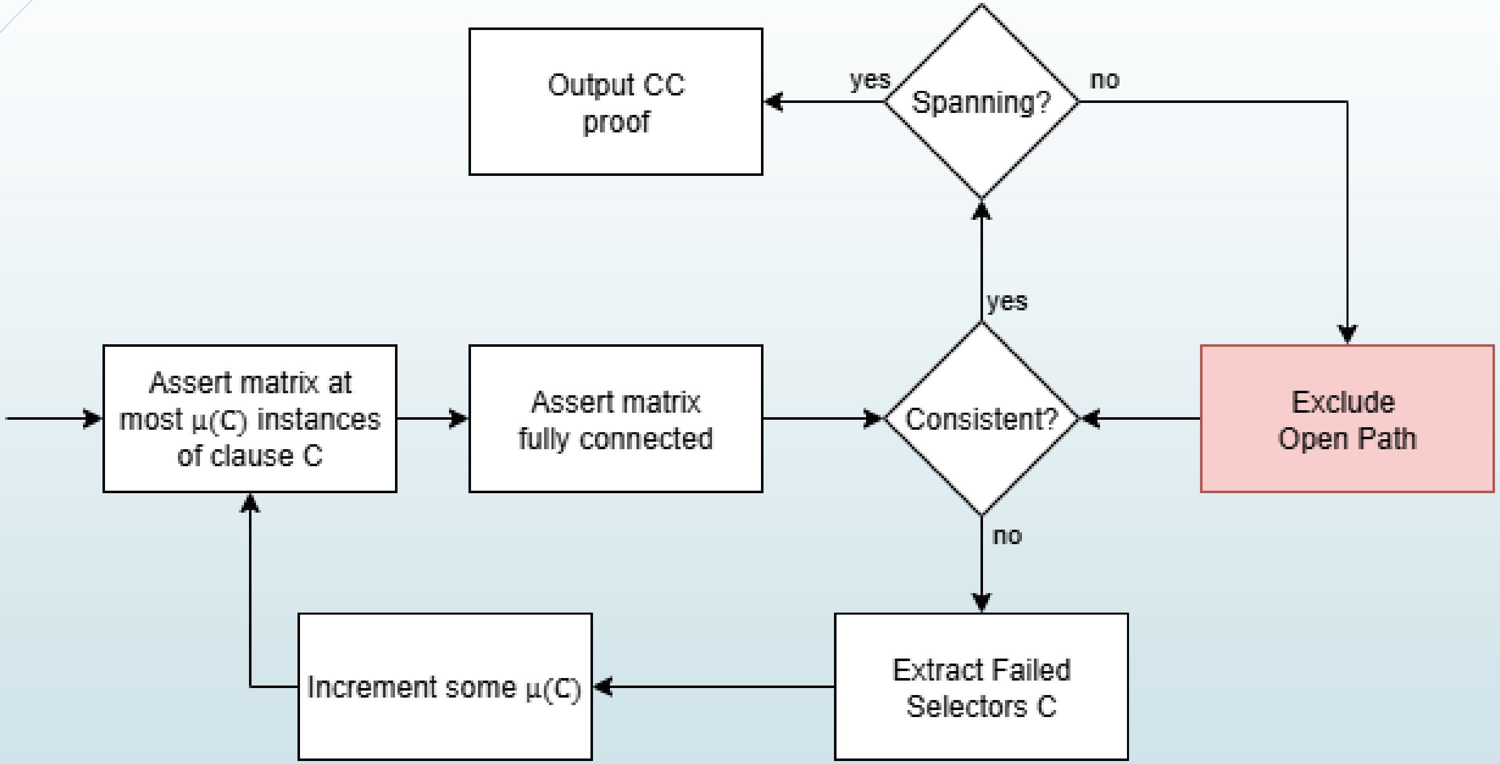
$P(f(x_1))$ $\qquad$ $P(f(x_2))$

$$\mu\big(P(a)\big) \mapsto 1$$
$$\mu\Big(P(x) \vee \neg P\big(f(x)\big)\Big) \mapsto 2$$
$$\mu\big(\neg P(a)\big) \mapsto 0$$

Dynamic Matrix Sizing Idea



Dynamic Matrix Sizing Idea



- Eliminating open paths is **broken** now

Why Broken?

Why Broken?

- The **at most** – instead of exactly – breaks it!

Why Broken?

- The **at most** – instead of exactly – breaks it!
- Let‘s consider a propositional example

Why Broken?

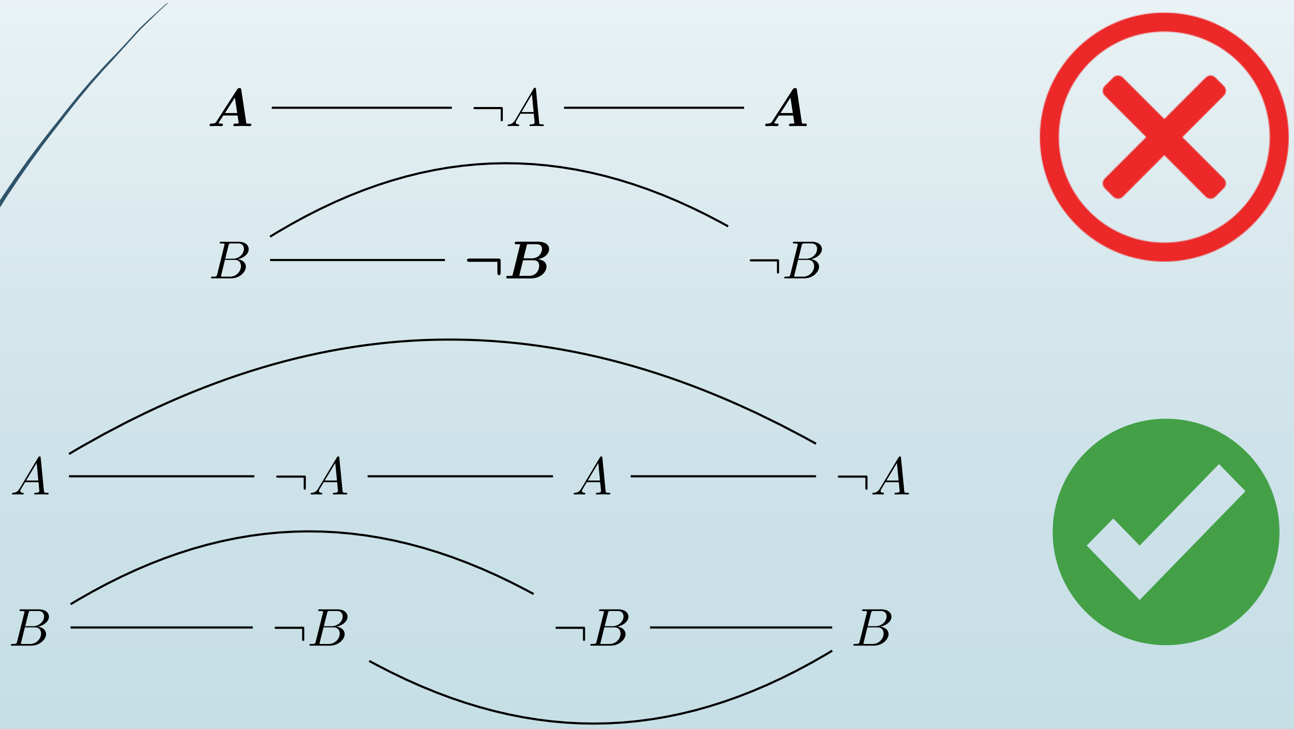
- The **at most** – instead of exactly – breaks it!
- Let‘s consider a propositional example

$$\{ A \lor B, \quad \neg A \lor \neg B, \quad A \lor \neg B, \quad \neg A \lor B \}$$

Why Broken?

- The **at most** – instead of exactly – breaks it!
- Let's consider a propositional example

$\{ A \vee B, \quad \neg A \vee \neg B, \quad A \vee \neg B, \quad \neg A \vee B \}$

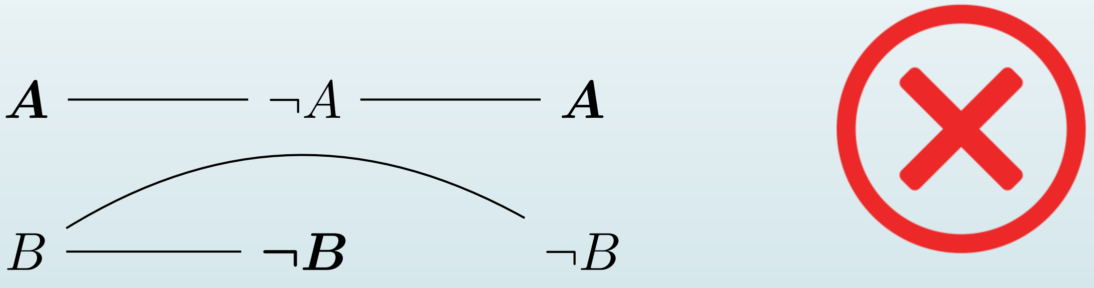


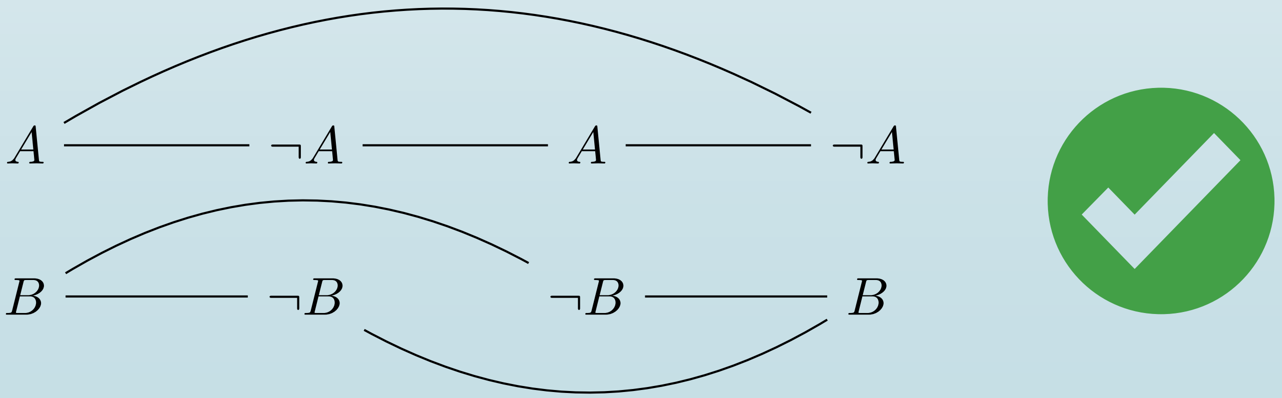
Why Broken?

$$\bigwedge_{S\in\bar{S}} S \Rightarrow \bigvee_{\{L,K\}\subseteq U} \langle L \sim K \rangle$$

- The **at most** – instead of exactly – breaks it!
- Let‘s consider a propositional example

{ A ∨ B, ¬A ∨ ¬B, A ∨ ¬B, ¬A ∨ B }



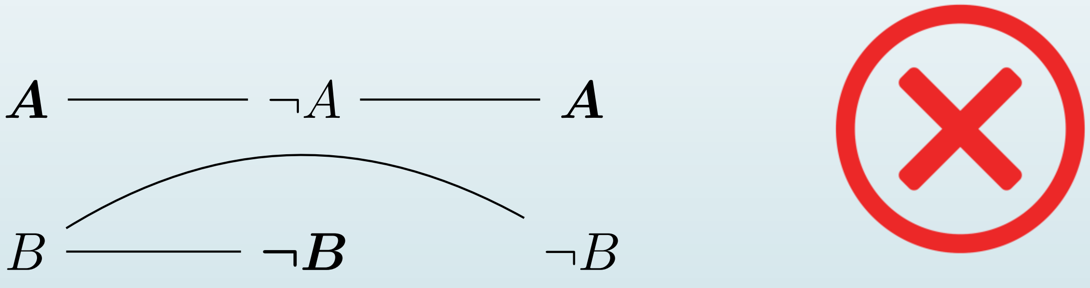


Why Broken?

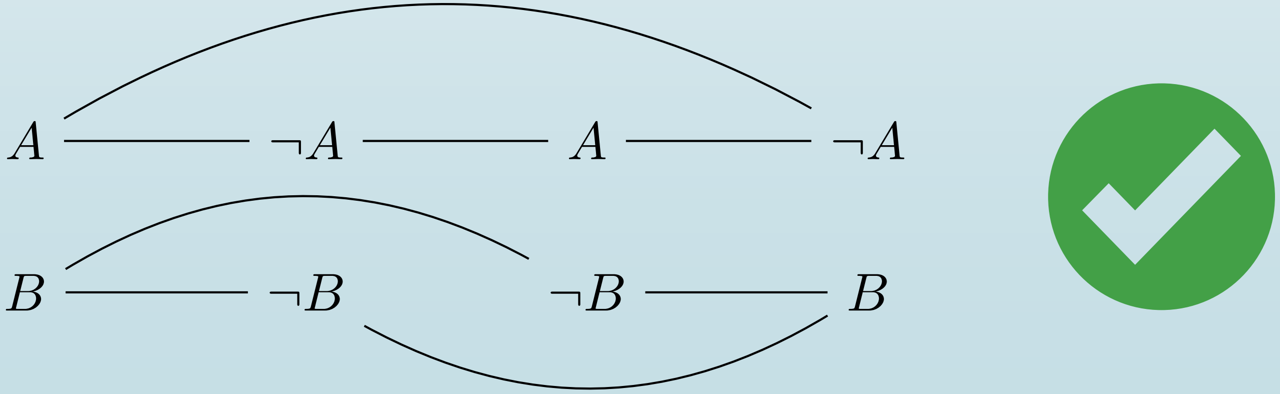
$$\bigwedge_{S\in\bar{S}} S \Rightarrow \bigvee_{\{L,K\}\subseteq U} \langle L \sim K\rangle$$

- The **at most** – instead of exactly – breaks it!
- Let‘s consider a propositional example

$\{ A \vee B, \quad \neg A \vee \neg B, \quad A \vee \neg B, \quad \neg A \vee B \}$



$$(S^1_{A\vee B} \wedge S^1_{\neg A\vee\neg B} \wedge S^1_{A\vee\neg B}) \Rightarrow \bot$$



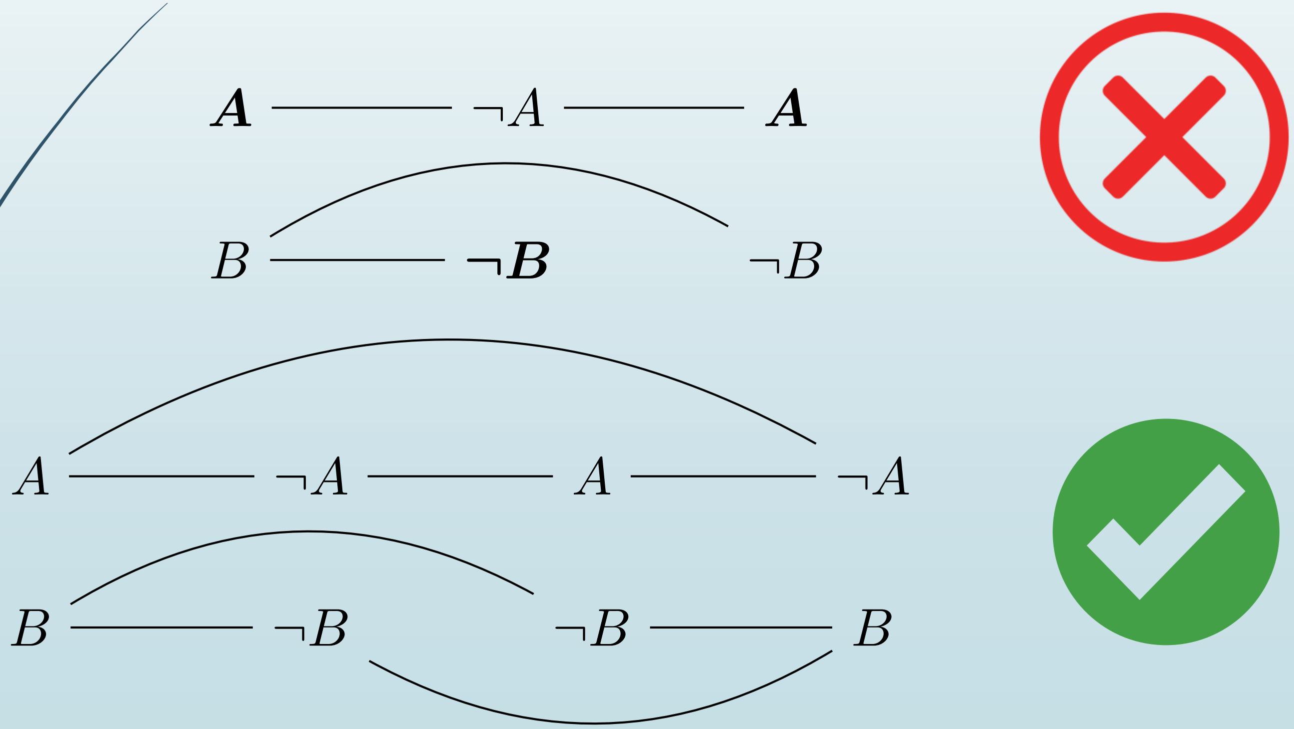
Solution

- We have to choose another element not yet in the matrix

Solution

- We have to choose another element not yet in the matrix

$$\{ A \vee B, \quad \neg A \vee \neg B, \quad A \vee \neg B, \quad \neg A \vee B \}$$

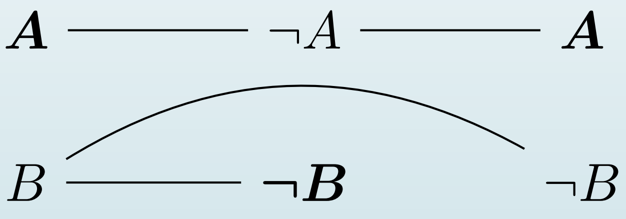


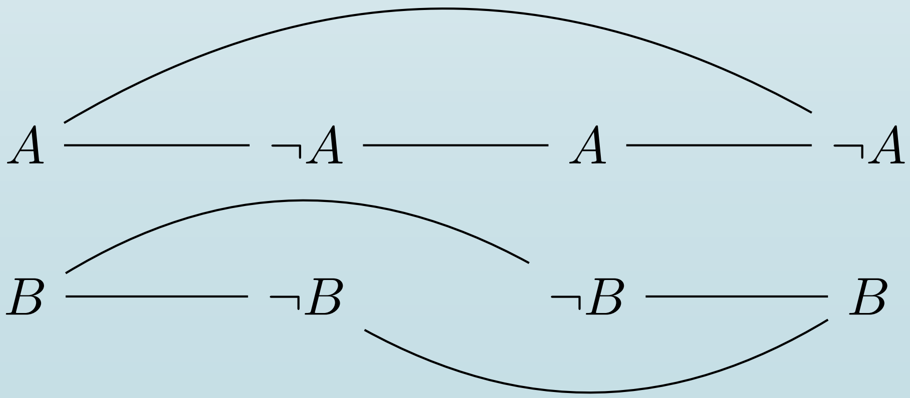
Solution

$$\bigwedge_{S\in\bar{S}} S \Rightarrow \bigvee_{\{L,K\}\subseteq U} \langle L \sim K\rangle \vee \bigvee_{L\in U} F_L$$

- We have to choose another element not yet in the matrix

$\{ A \vee B, \quad \neg A \vee \neg B, \quad A \vee \neg B, \quad \neg A \vee B \}$



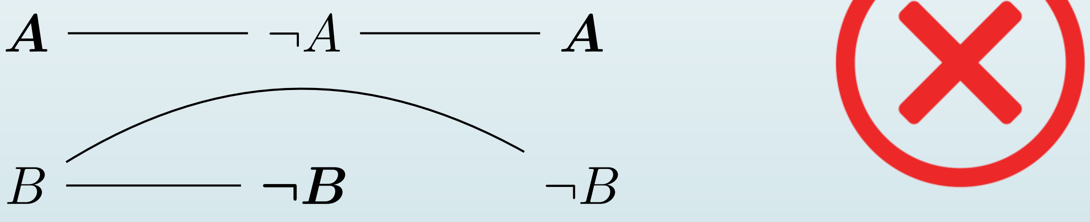


Solution

$$\bigwedge_{S\in\bar{S}} S \Rightarrow \bigvee_{\{L,K\}\subseteq U} \langle L \sim K\rangle \vee \bigvee_{L\in U} F_L$$

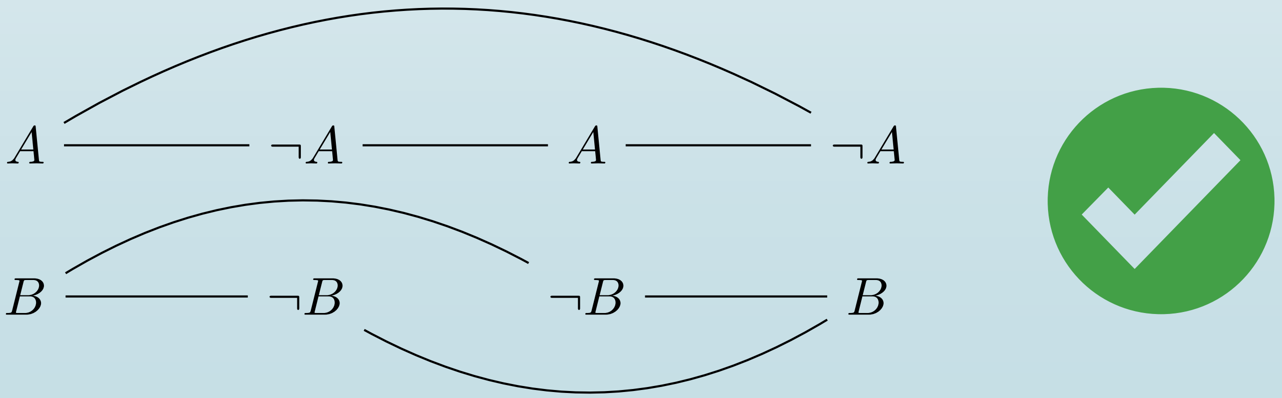
- We have to choose another element not yet in the matrix

$\{ A \vee B, \quad \neg A \vee \neg B, \quad A \vee \neg B, \quad \neg A \vee B \}$



$$(S^1_{A\vee B} \wedge S^1_{\neg A\vee\neg B} \wedge S^1_{A\vee\neg B}) \Rightarrow$$

$$(S^2_{A\vee B} \vee S^2_{\neg A\vee\neg B} \vee S^2_{A\vee\neg B} \vee S^1_{\neg A\vee B})$$



Optimisations

Optimisations

- Monotonicity of selectors of some clause C

$$S_C^{i+1} \Rightarrow S_C^{i}$$

Optimisations

- Monotonicity of selectors of some clause C

$$S_C^{i+1} \Rightarrow S_C^i$$

- Term Ordering
  - If $i < j$ then $C^i \nprec C^j$ modulo unifier
  - e.g., assume term order $\mathrm{a} \prec \mathrm{f(a)}$ and some matrix containing $P(x_1)$ and $P(x_2)$
  - Having $\sigma(x_1) \mapsto \mathrm{f(a)}$ forbids $\sigma(x_2) \mapsto a$

Optimisations

- Monotonicity of selectors of some clause C

$$S_C^{i+1} \Rightarrow S_C^{i}$$

- Term Ordering
  - If $i < j$ then $C^i \nprec C^j$ modulo unifier
  - e.g., assume term order $\mathrm{a} \prec \mathrm{f(a)}$ and some matrix containing $P(x_1)$ and $P(x_2)$
  - Having $\sigma(x_1) \mapsto \mathrm{f(a)}$ forbids $\sigma(x_2) \mapsto a$
- "Subsumption"
  - For any clause instances C^i and D^j we have $C^i \neq D^j$ modulo unifier

Optimisations

- Monotonicity of selectors of some clause C

$$S_C^{i+1} \Rightarrow S_C^i$$

- Term Ordering
  - If $i < j$ then $C^i \nprec C^j$ modulo unifier
  - e.g., assume term order $\mathrm{a} \prec \mathrm{f(a)}$ and some matrix containing $P(x_1)$ and $P(x_2)$
  - Having $\sigma(x_1) \mapsto \mathrm{f(a)}$ forbids $\sigma(x_2) \mapsto a$
- "Subsumption"
  - For any clause instances C^i and D^j we have $C^i \neq D^j$ modulo unifier
- ("AVATAR"-like) Clause splitting

Optimisations

- Monotonicity of selectors of some clause C

$$S_C^{i+1} \Rightarrow S_C^i$$

- Term Ordering
  - If $i < j$ then $C^i \nprec C^j$ modulo unifier
  - e.g., assume term order $\mathrm{a} \prec \mathrm{f(a)}$ and some matrix containing $P(x_1)$ and $P(x_2)$
  - Having $\sigma(x_1) \mapsto \mathrm{f(a)}$ forbids $\sigma(x_2) \mapsto a$
- "Subsumption"
  - For any clause instances C^i and D^j we have $C^i \neq D^j$ modulo unifier
- ("AVATAR"-like) Clause splitting
- Decision procedure for EPR ("Bernays-Schönfinkel") fragment

Results

- Prototype *UpCoP* using both *CaDiCal* (SAT) and *Z3* (SMT) backend
- Compared against *meanCoP* (complete mode)

Results

- Prototype *UpCoP* using both *CaDiCal* (SAT) and *Z3* (SMT) backend
- Compared against *meanCoP* (complete mode)

Solver	$\text{UpCoP}_{\text{SMT}}$	$\text{UpCoP}_{\text{SAT}}$	$\text{UpCoP}_{\text{SMT}}$	$\text{UpCoP}_{\text{SAT}}$	$\text{UpCoP}_{\text{SMT}}$	$\text{UpCoP}_{\text{SAT}}$	meanCoP
Enc.	$\mathcal{E}_{\text{M}}$		$\mathcal{E}_{\text{U}}$		$\mathcal{E}_{\text{H}}$		
Solved	928	855	1152	1055	1272	1264	1972
Unique	27	20	109	93	105	76	551

Results

- Prototype *UpCoP* using both *CaDiCal* (SAT) and *Z3* (SMT) backend
- Compared against *meanCoP* (complete mode)

Solver	UpCoP$_{SMT}$	UpCoP$_{SAT}$	UpCoP$_{SMT}$	UpCoP$_{SAT}$	UpCoP$_{SMT}$	UpCoP$_{SAT}$	meanCoP
Enc.	$\mathcal{E}_M$		$\mathcal{E}_U$		$\mathcal{E}_H$		
Solved	928	855	1152	1055	1272	1264	1972
Unique	27	20	109	93	105	76	551

Huch! What went wrong??

Several Reasons for Bad Performance

Several Reasons for Bad Performance

- Overhead of delegating reasoning to SAT core

Several Reasons for Bad Performance

- Overhead of delegating reasoning to SAT core
- Learned conflicts often unhelpful/very specific

Several Reasons for Bad Performance

- Overhead of delegating reasoning to SAT core
- Learned conflicts often unhelpful/very specific
- The SAT solver "builds" independent matrix parts

Several Reasons for Bad Performance

- Overhead of delegating reasoning to SAT core
- Learned conflicts often unhelpful/very specific
- The SAT solver "builds" independent matrix parts

$$\{\, P(a), \qquad \neg Q(x) \vee Q\big(f(x)\big), \qquad \neg P(a) \,\}$$

Several Reasons for Bad Performance

- Overhead of delegating reasoning to SAT core
- Learned conflicts often unhelpful/very specific
- The SAT solver "builds" independent matrix parts

$$\{\, P(a), \quad \neg Q(x) \vee Q(f(x)), \quad \neg P(a) \,\}$$

$$\boxed{P(a)} \qquad \boxed{\begin{array}{ccc} \neg Q(x_1) & \neg Q(x_2) & \neg Q(x_3) \\ Q(f(x_1)) & Q(f(x_2)) & Q(f(x_3)) \end{array}}$$

Summary

Summary

- Encoded the **existence of a spanning matrix** as a SAT problem
  - Usually: propositionally unsatisfiable ➔ FO proof

Summary

- Encoded the **existence of a spanning matrix** as a SAT problem
  - Usually: propositionally unsatisfiable ➔ FO proof
- Evaluated our prototype using a SAT and SMT backend

Summary

- Encoded the **existence of a spanning matrix** as a SAT problem
  - Usually: propositionally unsatisfiable ➔ FO proof
- Evaluated our prototype using a SAT and SMT backend
  - Experimental results are rather modest

Summary

- Encoded the **existence of a spanning matrix** as a SAT problem
  - Usually: propositionally unsatisfiable ➔ FO proof
- Evaluated our prototype using a SAT and SMT backend
  - Experimental results are rather modest

In-depth investigation of SAT for CC reasoning

Summary

- Encoded the **existence of a spanning matrix** as a SAT problem
  - Usually: propositionally unsatisfiable ➔ FO proof
- Evaluated our prototype using a SAT and SMT backend
  - Experimental results are rather modest

In-depth investigation of SAT for CC reasoning

Experiments:
No SAT-guided CC ☹

Summary

- Encoded ~~the **existence of a spanning matrix**~~ what is definitely unhelpful towards a closed tableaux as a "SAT" problem
  - Usually: propositionally unsatisfiable ➔ FO proof
- Evaluated our prototype using a SAT and SMT backend
  - Experimental results are rather modest

In-depth investigation of SAT for CC reasoning

~~**Experiments: No SAT-guided CC ☹**~~

Summary

what is definitely unhelpful towards a closed tableaux

- Encoded ~~the **existence of a spanning matrix**~~ as a "SAT" problem
- Ev … nd

Stay tuned for the next lecture ☺

In-depth investigation of SAT for CC reasoning

Experiments: No SAT-guided CC ☹

Summary

what is definitely unhelpful towards a closed tableaux

- Encoded ~~the **existence of a spanning matrix**~~ as a "SAT" pro
  -
- Ev … nd
  -

Stay tuned for the next lecture ☺

~~Experiments: No SAT-guided CC ☹~~

In-depth investigation of SAT for CC reasoning

Thanks for your attention!

Summary

- Encoded ~~the existence of a spanning matrix~~ what is definitely unhelpful towards a closed tableaux as a "SAT" pro
- Ev … nd

In-depth investigation of SAT for CC reasoning

Stay tuned for the next lecture ☺

~~Experiments: No SAT-guided CC ☹~~

Thanks for your attention!

Questions?