Interaktive Extraktion minimaler unerfüllbarer Kerne unter Verwendung von Metalernen (Diplomarbeit)

Johannes Dellert

Universität Tübingen

08.03.2013

Übersicht

Minimale unerfüllbare Kerne

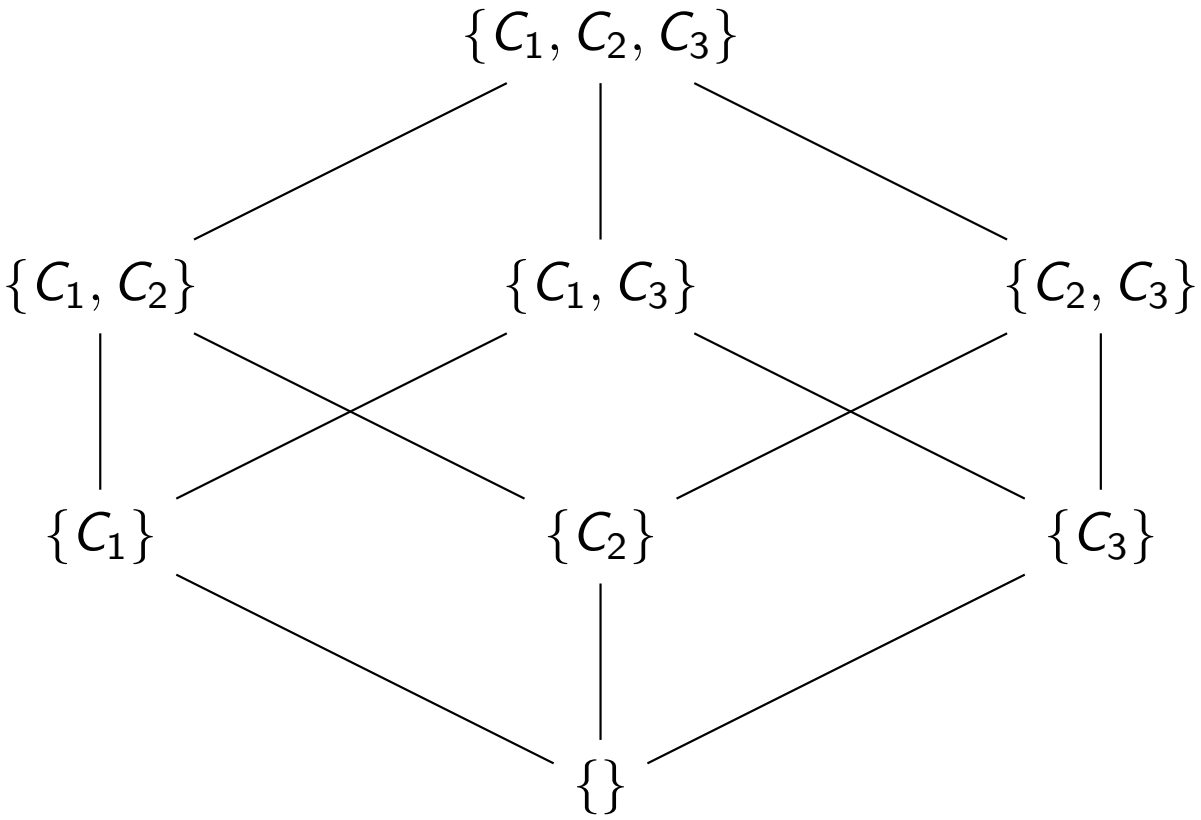
Definition

Eine Untermenge $F' \subseteq F$ einer unerfüllbaren Klauselmenge F ist eine **minimale unerfüllbare Untermenge (MUS)** von F wenn sie unerfüllbar, aber jede Untermenge $F'' \subset F'$ erfüllbar ist.

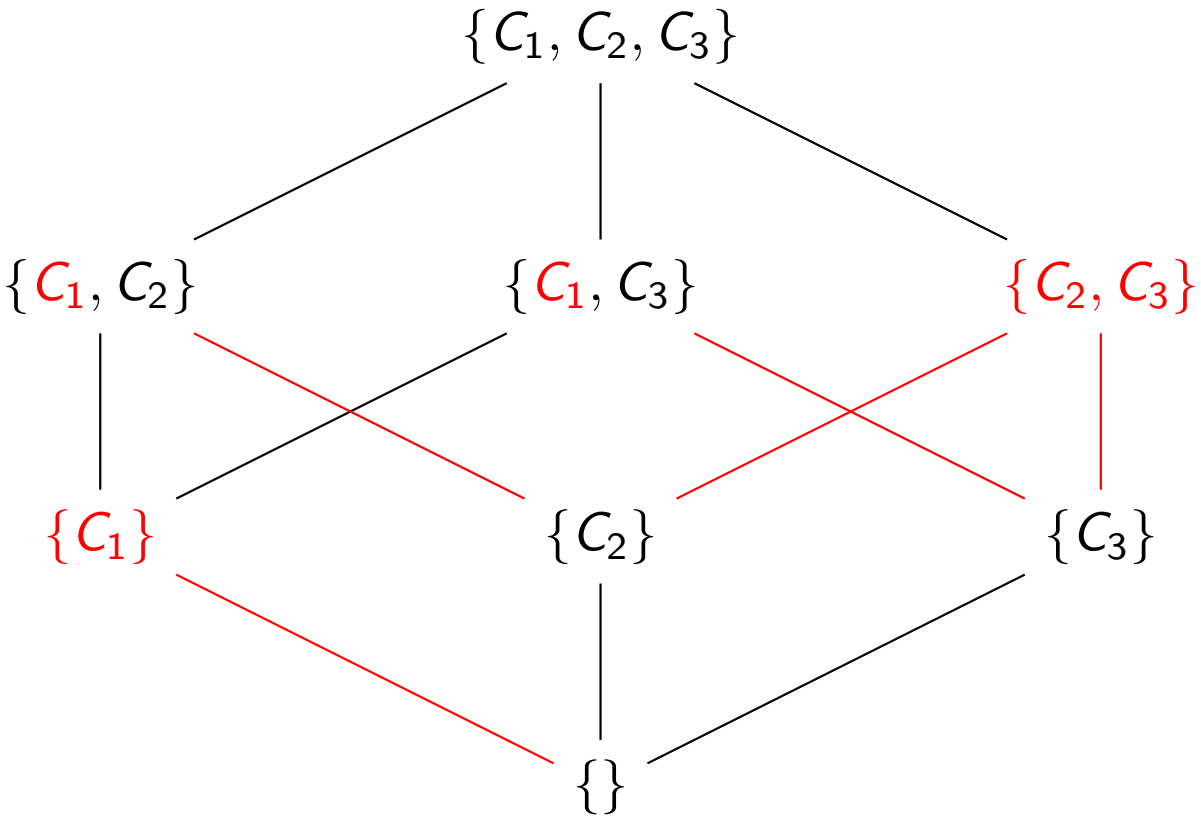
Eigenschaft

Eine Klauselmenge F ist eine MUS genau dann, wenn sie unerfüllbar ist, aber $F \backslash \{C\}$ erfüllbar ist für jede Klausel $C \in F$ (**kritische Klausel**).
(denn Konjunktion ist monoton bzgl. Unerfüllbarkeit)

MUSes im Teilmengenverband



MUSes im Teilmengenverband



Löschungsbasierter Extraktionsalgorithmus

Input: an unsatisfiable SAT instance $F = \{C_1, \ldots, C_m\}$ in CNF
Output: some minimal unsatisfiable subset $S \subseteq F$

1: $F' := \{\}$
2: **for** each clause $C_i \in F$ **do**
3: $\quad F' := F' \cup \{C_i \vee \neg s_i\}$ for a new selector variable s_i
4: **end for**
5: $\langle res, proof, \varphi, units \rangle := sat(F', \{s_1, \ldots, s_m\})$
6: **for** each $C_j \notin used\_clauses(proof)$ **do**
7: $\quad F' := F' \cup \{\{\neg s_j\}\}$
8: **end for**
9: $US := \{i \mid C_i \in used\_clauses(proof)\}$
10: $MUS := \{\}$
11: **while** (US is not empty) **do**
12: $\quad k :=$ select one index $\in US \setminus MUS$
13: $\quad US := US \setminus \{k\}$
14: $\quad \langle res, proof, \varphi, units \rangle := sat(F', \{s_i \mid i \in US \cup MUS\})$
15: $\quad$ **if** $res = sat$ **then**
16: $\quad\quad MUS := MUS \cup \{k\}$
17: $\quad$ **else**
18: $\quad\quad US := \{i \mid C_i \in used\_clauses(proof)\}$
19: $\quad\quad$ **for** each $C_j \notin used\_clauses(proof)$ **do**
20: $\quad\quad\quad F' := F' \cup \{\{\neg s_j\}\}$
21: $\quad\quad$ **end for**
22: $\quad$ **end if**
23: **end while**
24: **return** $S := \{C_i \mid i \in MUS\}$

Übersicht

Interaktive Extraktion: Motivation

Nachteile herkömmlicher Implementierungen von MUS-Extraktion

- MUS-Extraktoren sind Black Boxes, liefern irgendein MUS
- man erfährt nichts über den Suchraum, kann nur schwer vergleichen

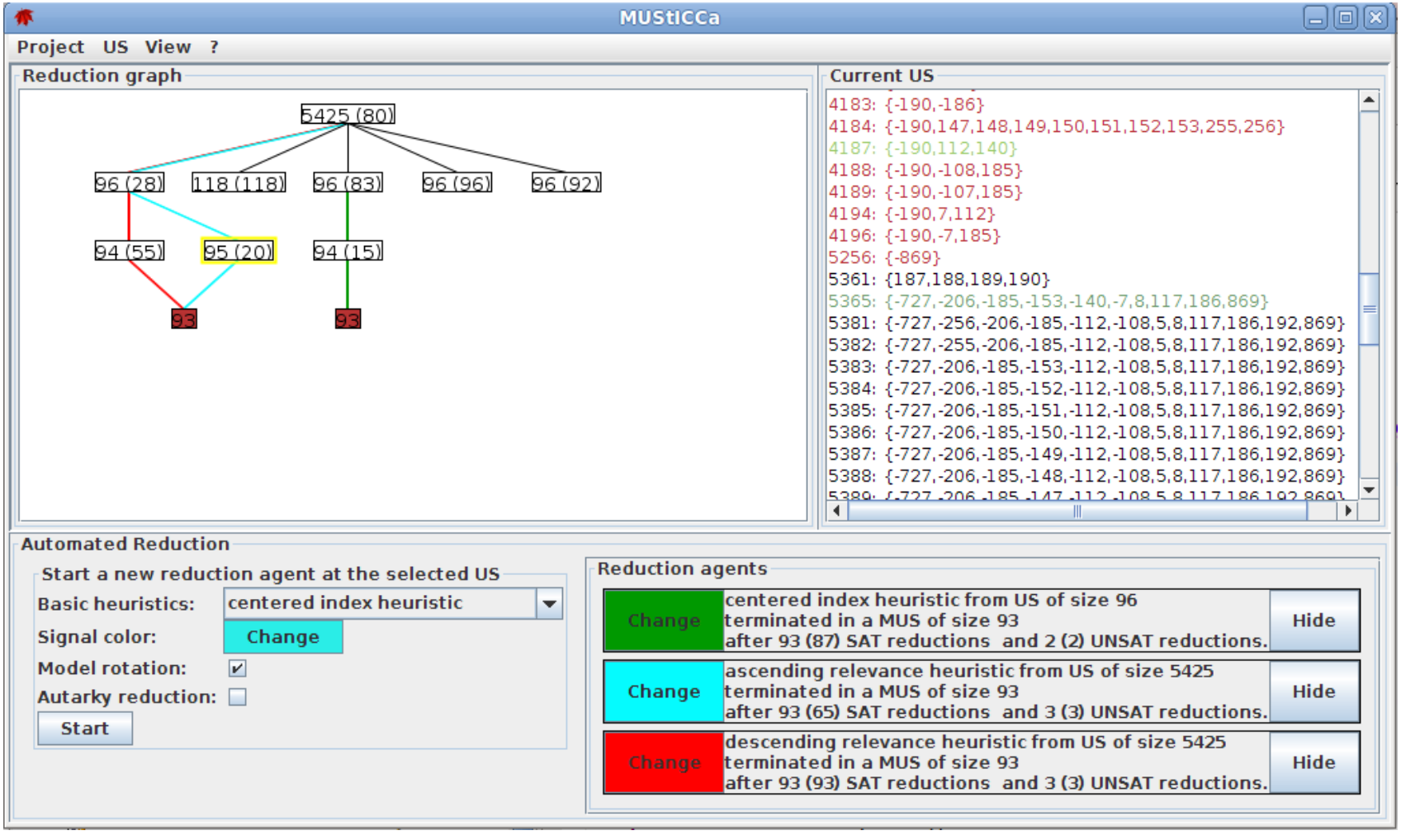
Idee: Interaktive Extraktion

- explizite Visualisierung des Suchraums als Teilhalbverband
- manuelle Ausführung vom Benutzer definierter Reduktionsschritte

Vorteile der interaktiven MUS-Extraktion

- liefert einen sehr guten Eindruck von der Struktur des Suchraums
- Experten können Reduktionsversuche anhand inhaltlicher Kriterien ausführen, mehrere interessante MUSes finden und vergleichen
- Synergieeffekte zwischen verschiedenen Zweigen, wenn partielles Wissen gespeichert wird und effizient abrufbar bleibt

Das Werkzeug MUStICCa



Reduktionsgraph: Konzept, Repräsentation

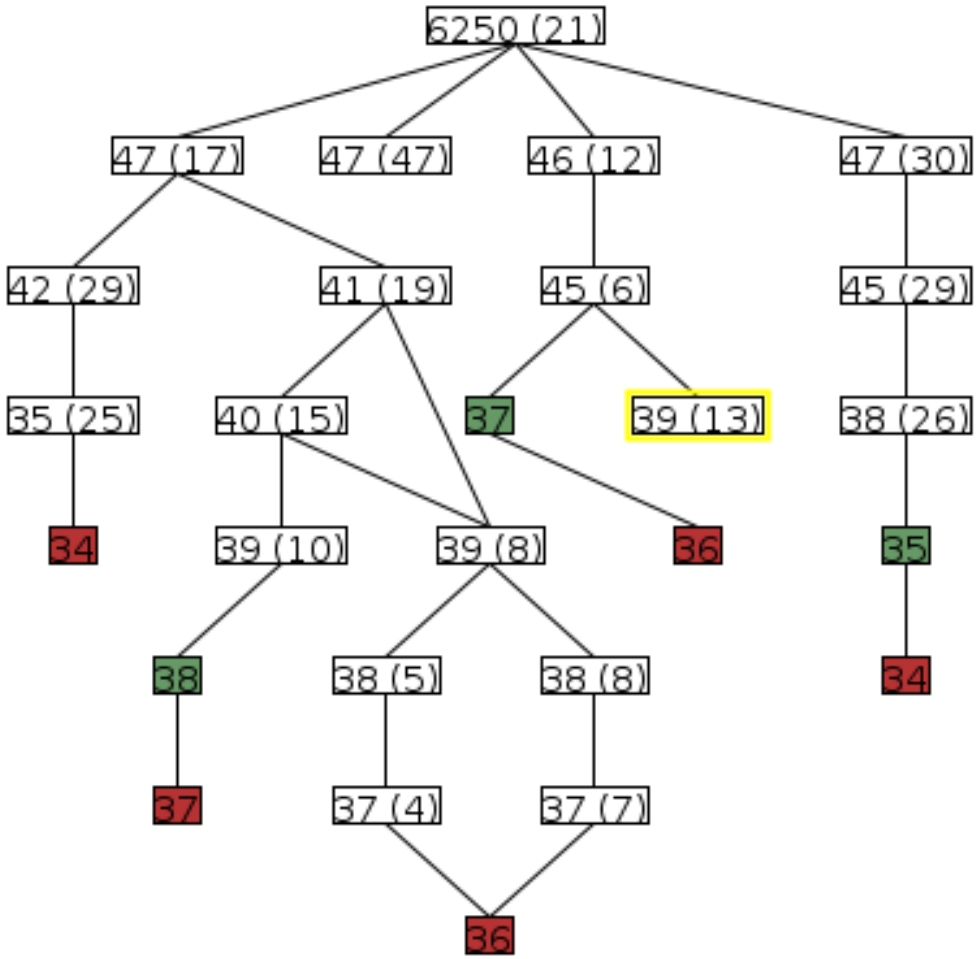
Definition

Sei F eine unerfüllbare Klauselmenge. Ein **Reduktionsgraph** $R = (V, E)$ für F besteht aus Knoten $v_i \in V$, die für unerfüllbare Untermengen $V_i \subseteq F$ stehen, und enthält eine Kante $(v_i, v_j) \in E$ gdw. V_j das Ergebnis der erfolgreichen Löschung einer redundanten Klausel aus V_i war, möglicherweise gefolgt von einer Zusatzoperation wie Klauselmengenverfeinerung.

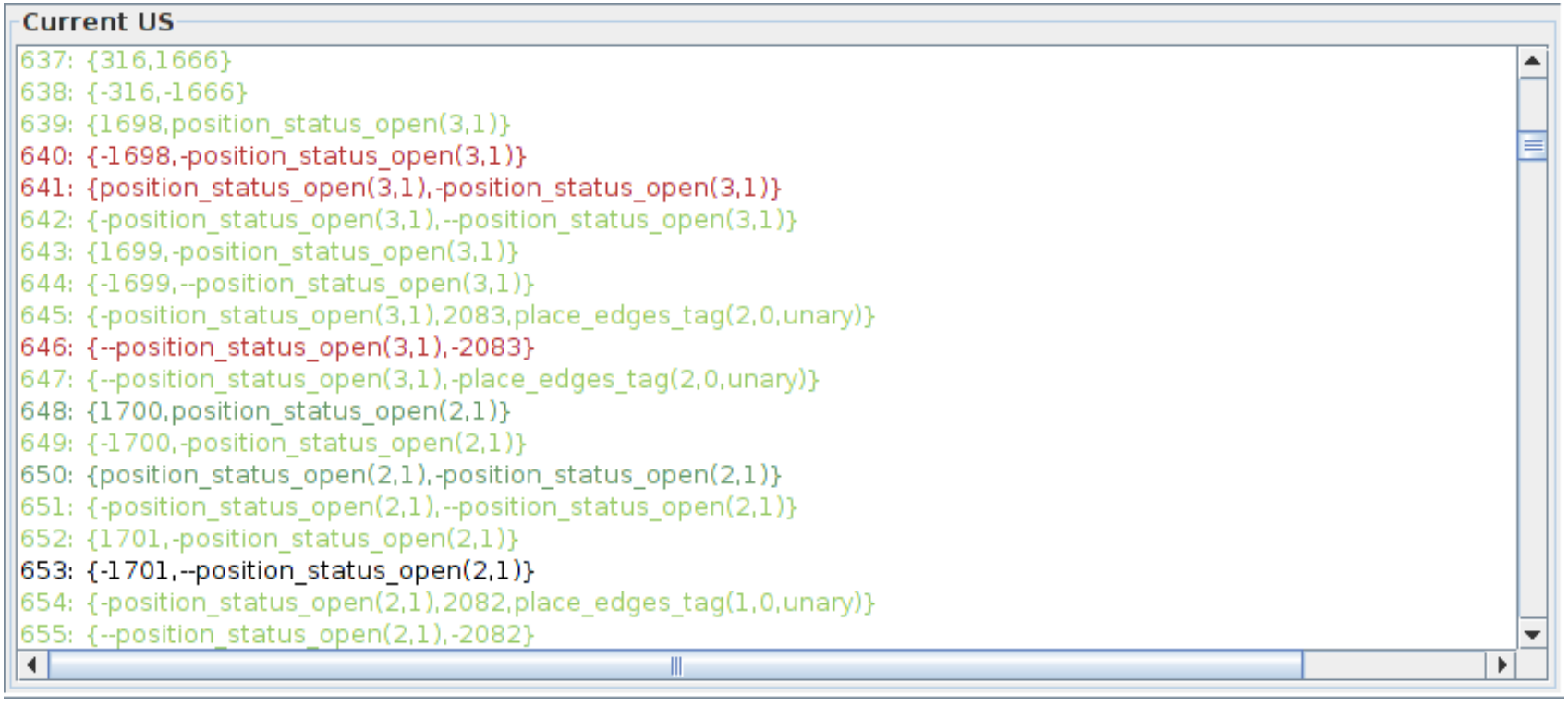
Repräsentation

- jede US im Reduktionsgraphen erhält eine ID, wird repräsentiert durch eine **Reduktionstabelle**, d.h. eine Abbildung der Klausel-IDs auf US-IDs, die die Kanten des Reduktionsgraphen darstellen
- spezielle Werte -1 für kritische Klausel, -2 für redundante Klausel mit unbekanntem Reduktionsergebnis, null für unbekannten Status

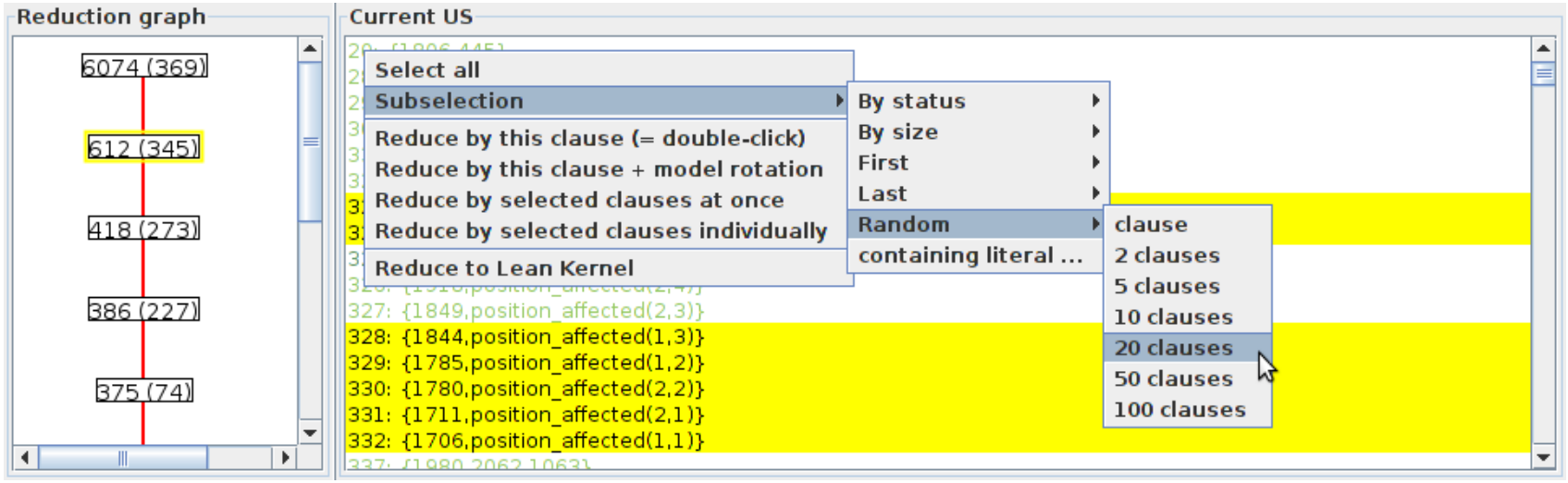
Reduktionsgraph: Visualisierung



US-Ansicht: Darstellung der US



US-Ansicht: Selektionsinterface



Das Selektionsinterface für Klauseln in der US

- funktioniert nach dem Filterprinzip (Definition einer Unterauswahl)
- ermöglicht (Unter-)Auswahl nach inhaltlichen Kriterien (Status, Größe, enthaltenes Literal)
- auch (Unter-)Auswahl einer festen Zahl von Klauseln möglich (vom Anfang, vom Ende, zufällig gewählt)

US-Ansicht: Funktionalität

Über US-Ansicht aufrufbare Funktionalität

- per Doppelklick wird ein einfacher Reduktionsversuch für die entsprechende Klausel ausgeführt
  - bei einem fehlgeschlagenen Versuch wird die festgestellte Kritikalität registriert und visualisiert
  - bei einem erfolgreichen Versuch wird die Ergebnis-US zum Reduktionsgraphen hinzugefügt (falls neu), die Reduktionstabelle wird aktualisiert, und wir wechseln zum verkleinerten US
- per Kontextmenü erreichbar sind
  - Versuch simultaner Reduktion aller momentan ausgewählter Klauseln
  - automatisierte Reduktionsversuche für alle ausgewählten Klauseln
  - Reduktionsversuch gefolgt von Modellrotation nach Scheitern
  - Reduktion der aktuellen US auf den Lean Kernel (Autarky Pruning)

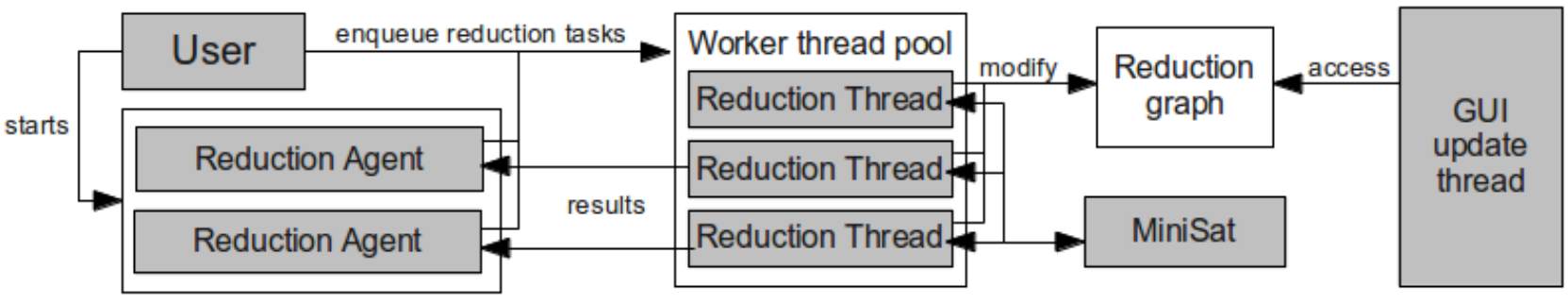
Automatisierte Extraktion: Konzepte

Reduktionsagenten

- sind durch den Benutzer definier- und startbar; agieren wie Durchläufe eines löschbasierten Extraktionsalgorithmus
- Reduktionsversuche mit bekanntem Ergebnis werden nur simuliert

Reduktionsthreads

- werden von einer Aufgabenverwaltung initialisiert und gestartet
- führen je einen Minisat-Aufruf aus und geben das Ergebnis per Callback an den jeweiligen Reduktionsagenten zurück



Automatisierte Extraktion: Heuristiken

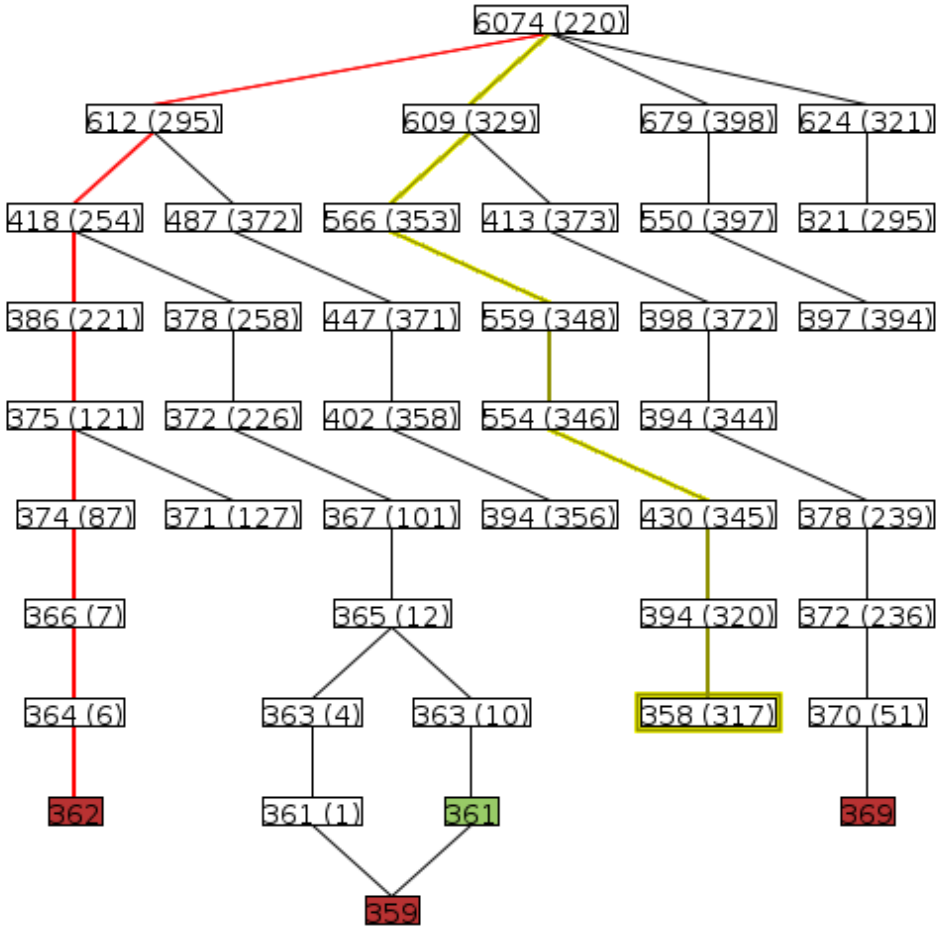
Reduktionsheuristik

- jeder Reduktionsagent erhält als Argument die Instanz einer **Reduktionsheuristik**, die ihn steuert
- eine Heuristik entscheidet anhand des aktuellen Wissens des Reduktionsagenten, welcher Reduktionsversuch als nächstes gestartet wird; sie stellt auch das Auffinden eines MUSes fest

Vordefinierte Heuristiken

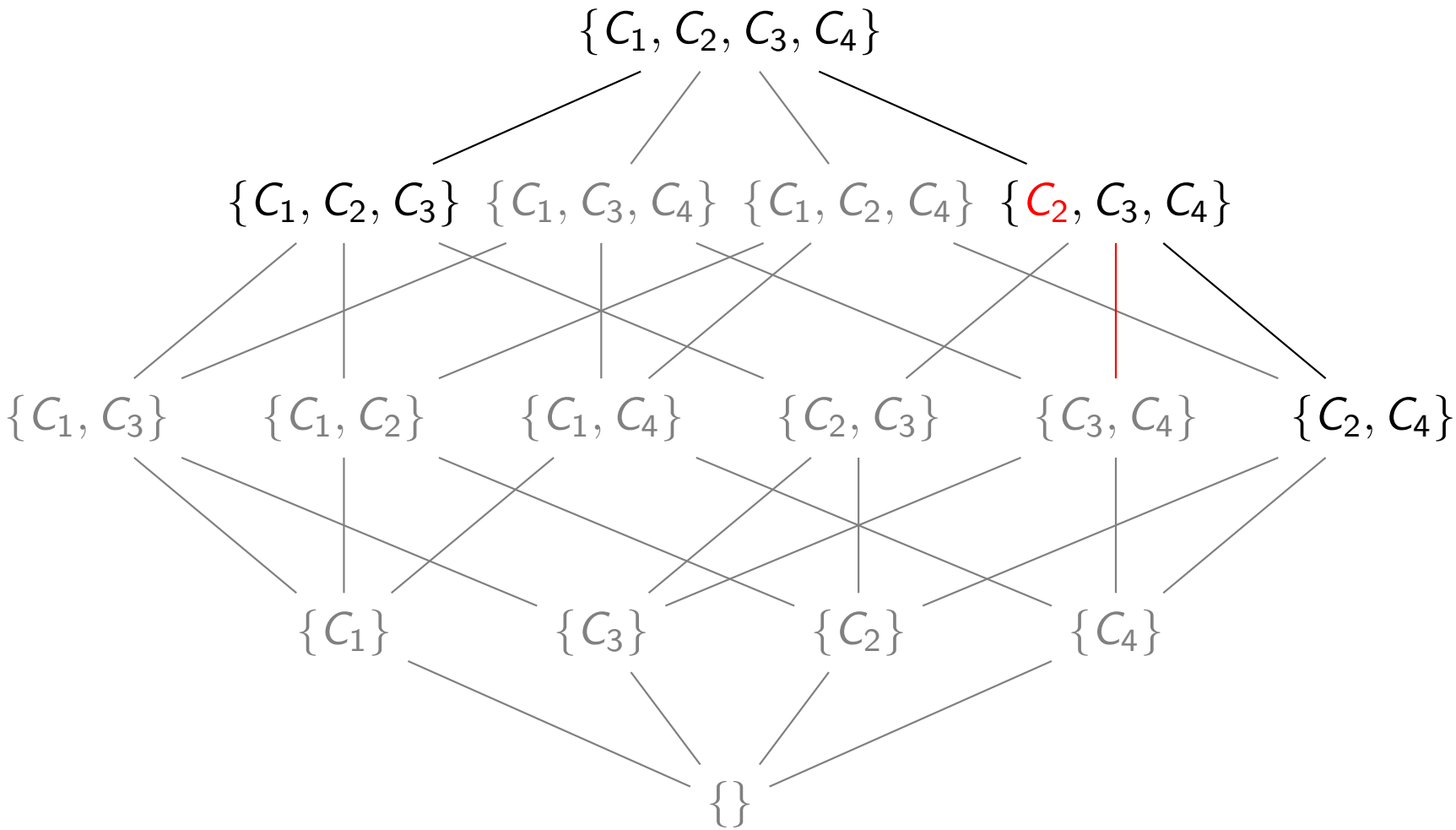
ascending index heuristic	goes through the US clauses by ascending ID
descending index heuristic	goes through the US clauses by descending ID
centered index heuristic	spirals through the US clauses starting in the middle
ascending relevance heuristic	goes through the US clauses in order of relevance, starting with the least relevant clause
descending relevance heuristic	goes through the US clauses in order of relevance, starting with the most relevant clause
centered relevance heuristic	starts with the US clauses of medium relevance, then spiraling out to ever more and less relevant ones

Automatisierte Extraktion: Agentenspuren

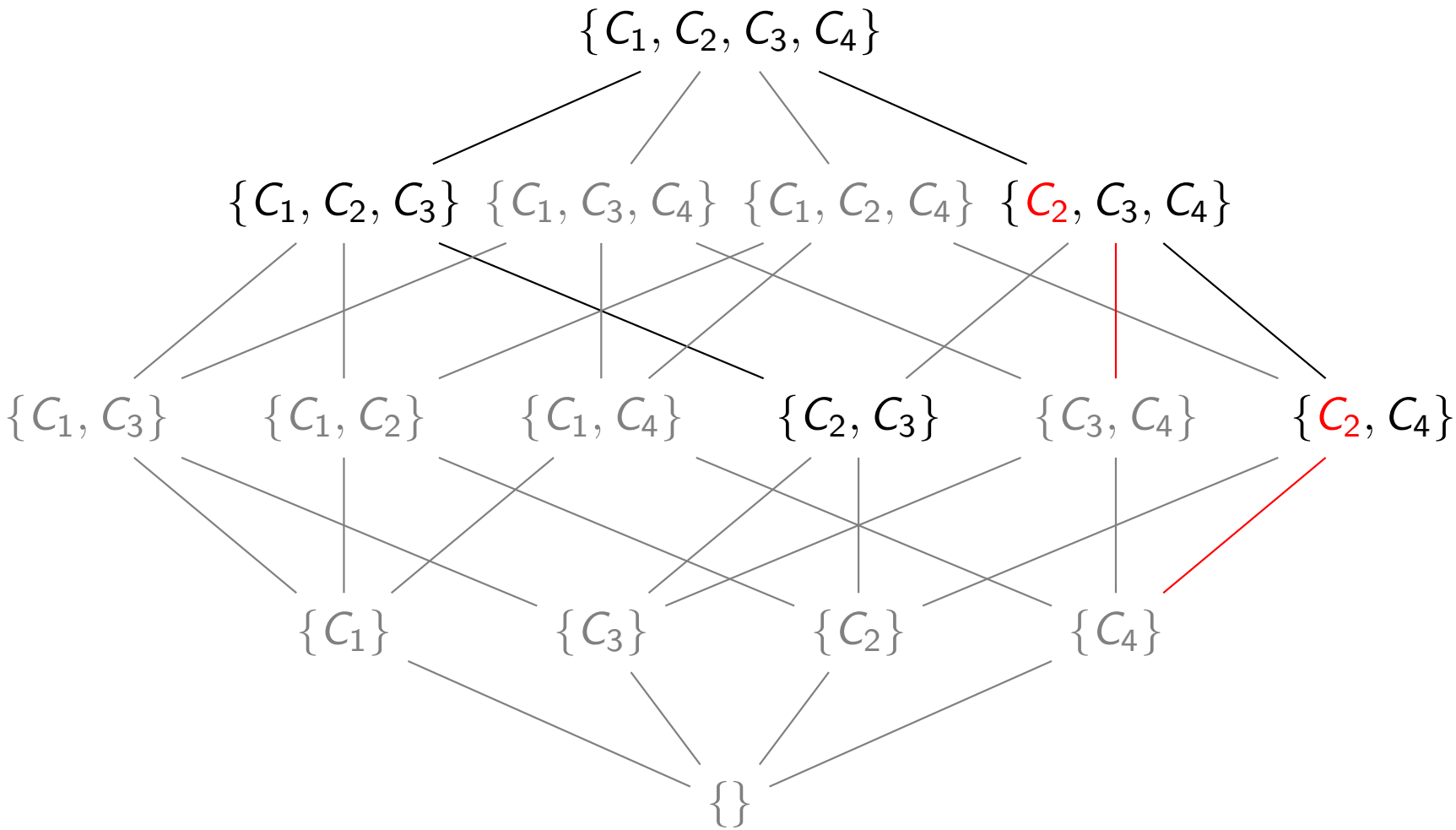


Übersicht

Metalernen: Motivation



Metalernen: Motivation



Metalernen: Motivation

Allgemeine Betrachtung des Problems

- es würde genügen, neue Information über die Kritikalität einer Klausel in einer US in alle Untermengen zu propagieren
- ein Reduktionsgraph stellt aber nicht alle Untermengenrelationen zwischen den durch seine Knoten repräsentierten Klauselmengen dar
- haben also nichttriviale Abhängigkeiten zwischen der Präsenz oder Abwesenheit bestimmter Kombinationen von Klauseln in einer US und der Kritikalität anderer Klauseln in dieser US
- wie können wir diese Abhängigkeiten systematisch modellieren und möglichst vollständig ausnutzen?

Metalernen: Grundidee

Zusammenhänge zwischen Selektorvariablen

- die Anwesenheit einer Klausel haben wir bisher schon durch positive Setzung der entsprechenden Selektorvariable modelliert
- setzen wir eine Selektorvariable auf falsch, so hat der entsprechende Constraint keine Wirkung mehr (entspricht Abwesenheit der Klausel)
- ist eine Klausel C kritisch in F, können wir ihre Präsenz durch Setzen der Selektorvariable s erzwingen; ist C in einem $F' \subset F$ kritisch, haben wir s unter den Bedingungen $\{\neg s_i \mid C_i \in F \backslash F'\}$

Idee: Metainstanz über Selektorvariablen

- wir erhalten aussagenlogische Constraints über Selektorvariablen, etliche davon sind Klauseln $\{s, s_1, ..., s_n\}$ (**Metaklauseln**)
- können diese zu einer **Metainstanz** $G = \{D_1, ..., D_n\}$ hinzufügen, um Wissen über nichttriviale Abhängigkeiten zu speichern

Lernen bei vergeblicher Löschung

Grundidee

- sind in US F', haben festgestellt, dass $F' \backslash \{C\}$ erfüllbar ist
- damit ist C kritisch in F' und allen seinen Untermengen

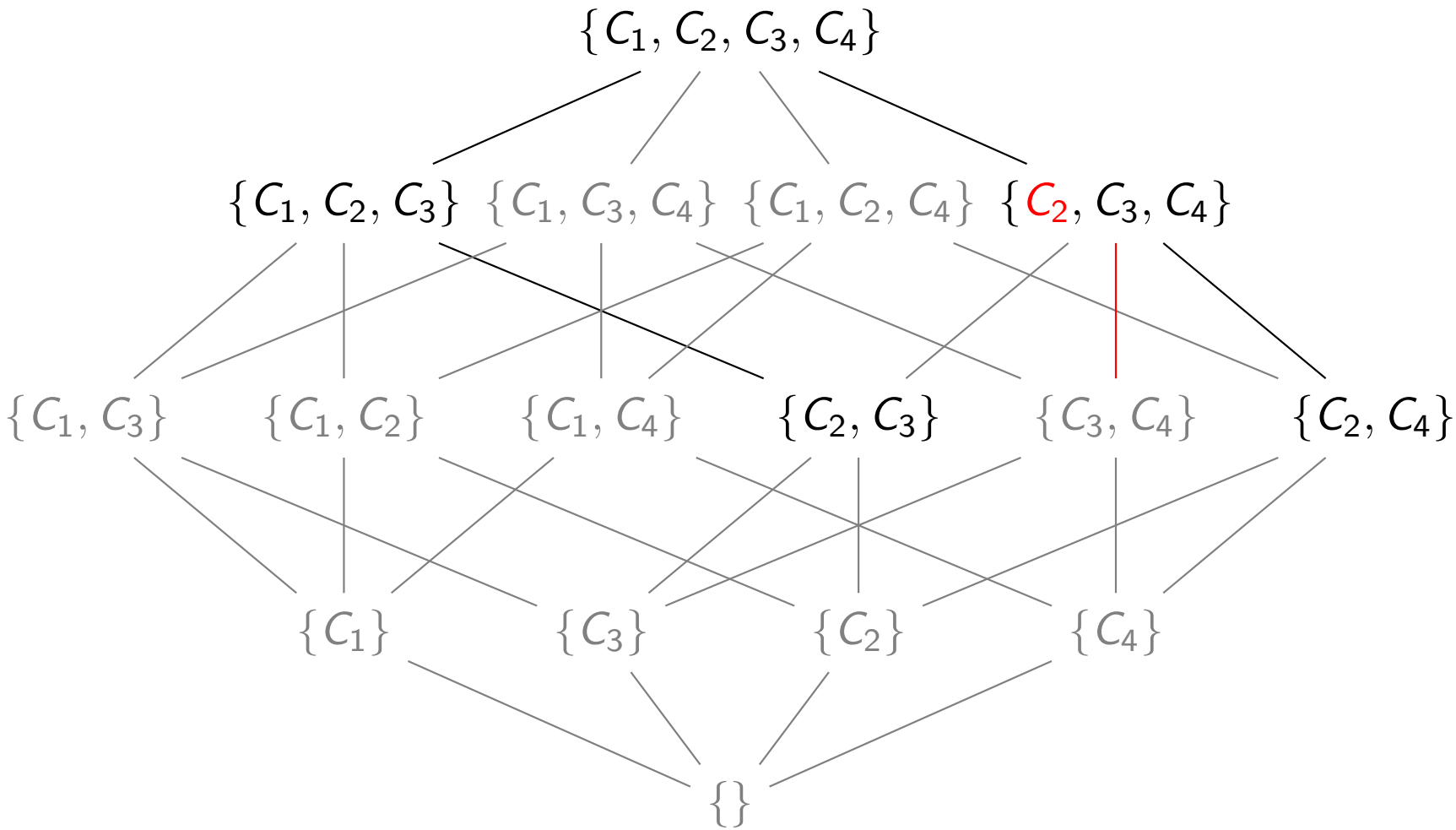
Definition

Ein **nach oben gerichteter Keil erfüllbarer Untermengen (SAT-Keil)** besteht aus einer erfüllbaren Untermenge $F' \subseteq F$ und allen ihren Untermengen, die dann ebenfalls erfüllbar sind.

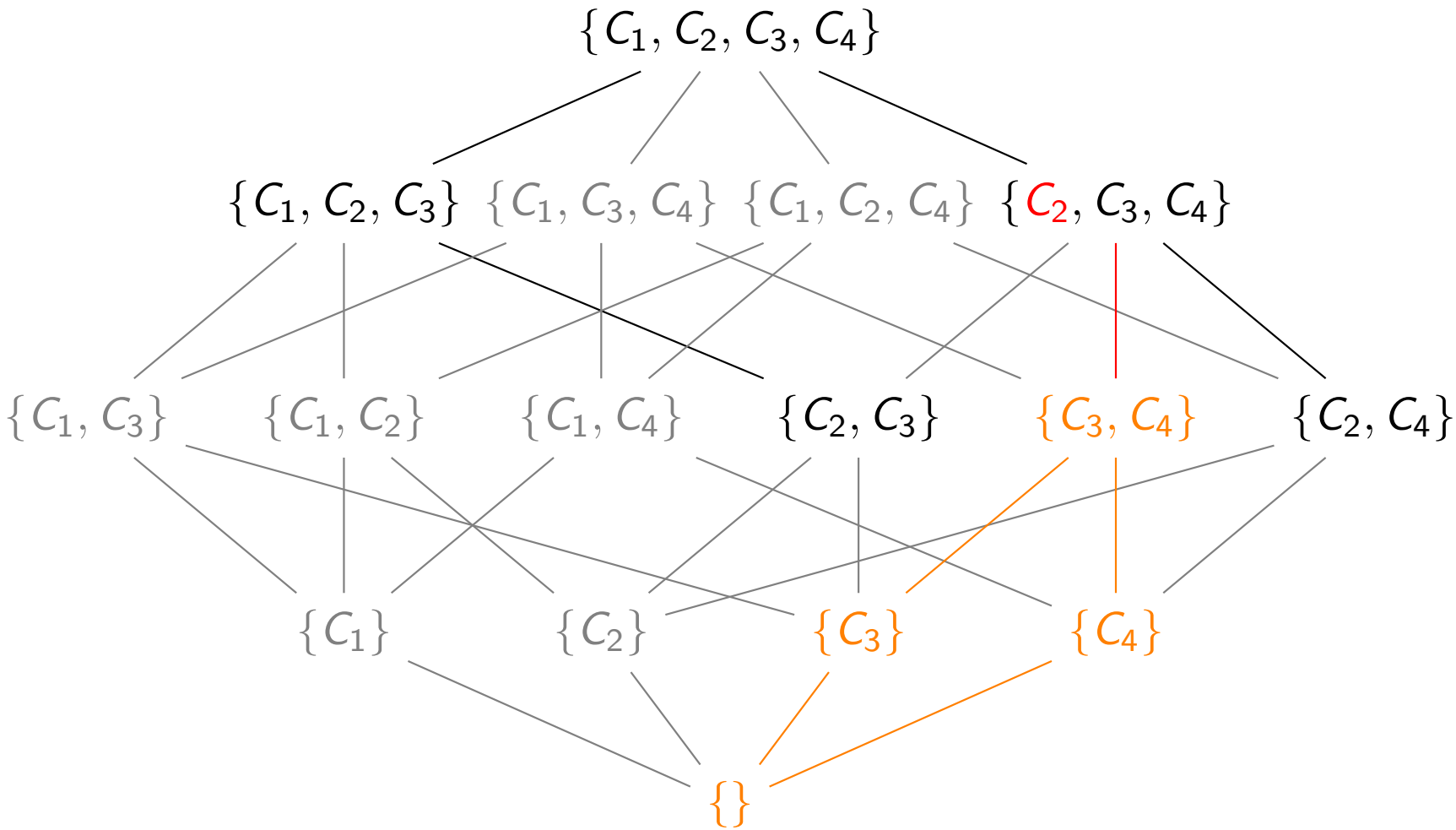
Repräsentation

- für $F \backslash F' = \{C_1, ..., C_n\}$ wissen wir jetzt $(s_1 \wedge ... \wedge s_n) \rightarrow s$
- das lässt sich schreiben als Klausel $\{s, s_1, ..., s_n\}$
- entspricht der Negation der Repräsentation des neuen SAT-Keils durch negative Selektorliterale

Beispiel: Lernen eines SAT-Keils



Beispiel: Lernen eines SAT-Keils



Kritikalitätspropagation

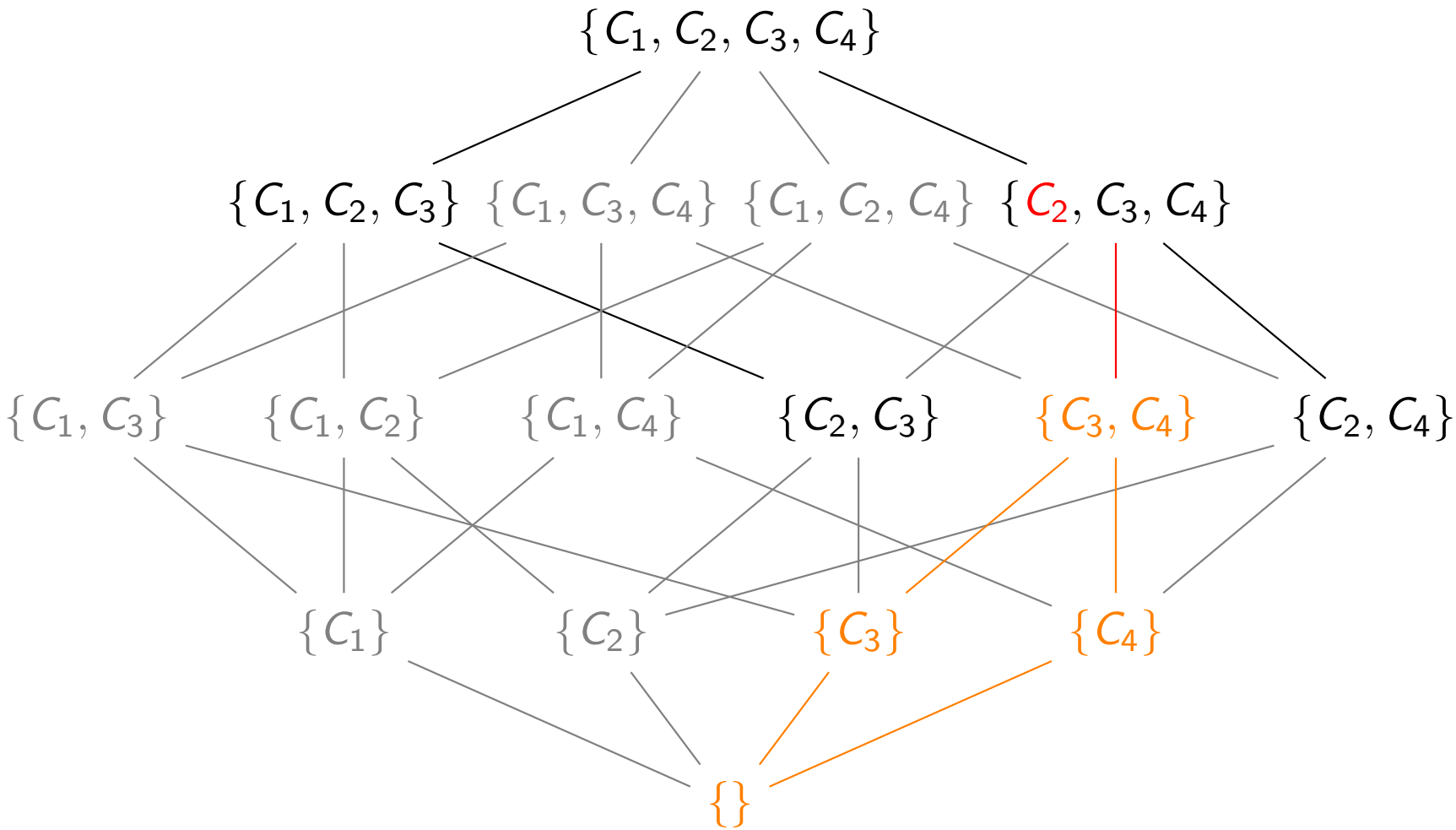
Prüfung einer Klauselmenge auf SAT-Keil-Mitgliedschaft

- repräsentiere die Klauselmenge durch Selektor-Units
- löse die Metainstanz mit den Selektor-Units als Annahmen; bei Ergebnis UNSAT ist die Klauselmenge in einem SAT-Keil

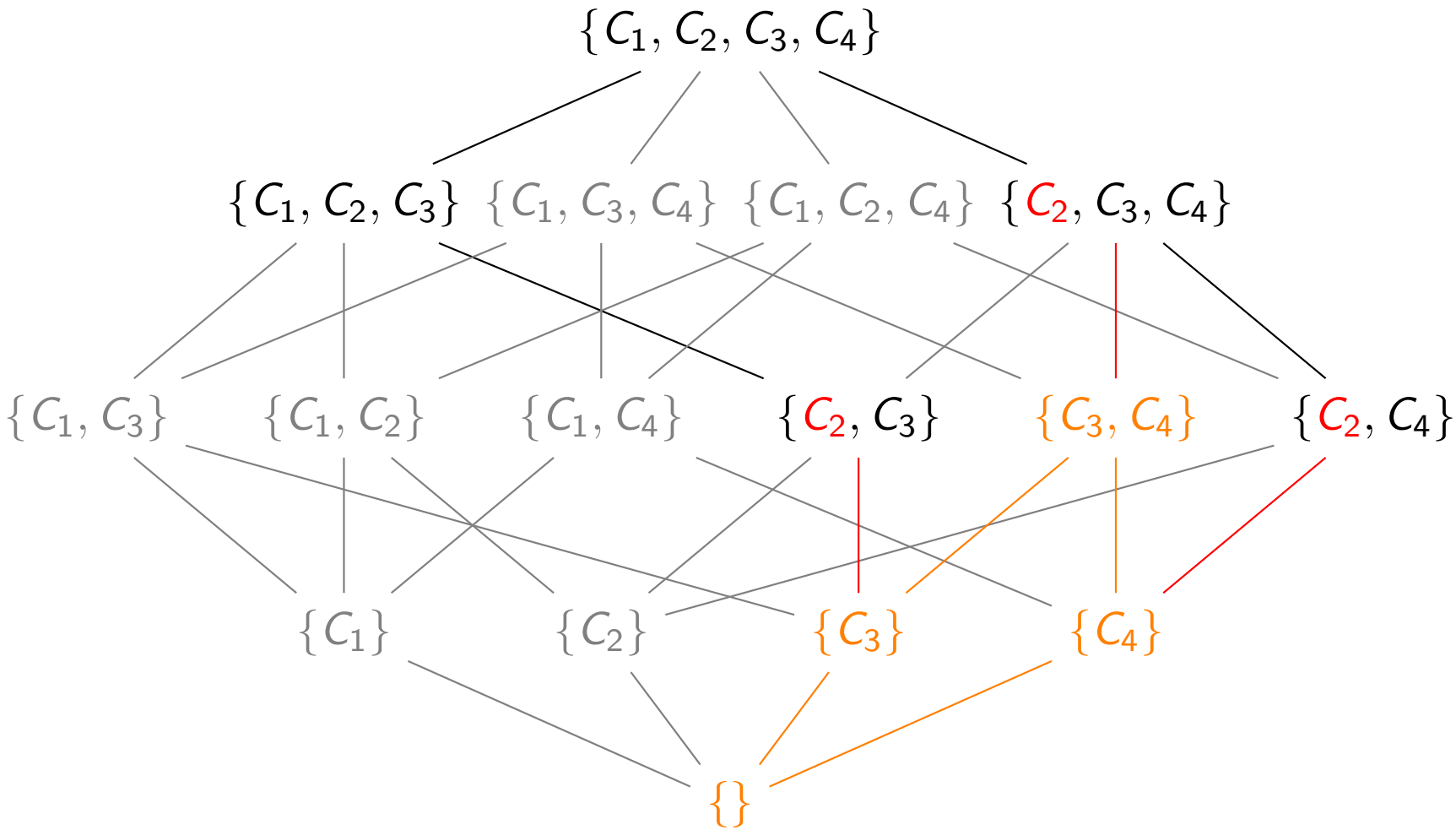
Idee: Kritikalitätspropagation

- wollen für eine gegebene US schnell alle kritischen Klauseln finden, aber nicht für jede Klausel einen SAT-Aufruf machen müssen
- es reicht, die Selektor-Units für die US durch das Meta-Problem zu propagieren; werden dabei positive Selektorliterale propagiert, so sind die entsprechenden Klauseln kritisch

Beispiel: Kritikalitätspropagation



Beispiel: Kritikalitätspropagation



Lernen bei erfolgreicher Löschung

Grundidee

- sind in US F', haben festgestellt, dass $F' \backslash \{C\}$ unerfüllbar ist
- damit ist C redundant in F' und allen seinen Obermengen

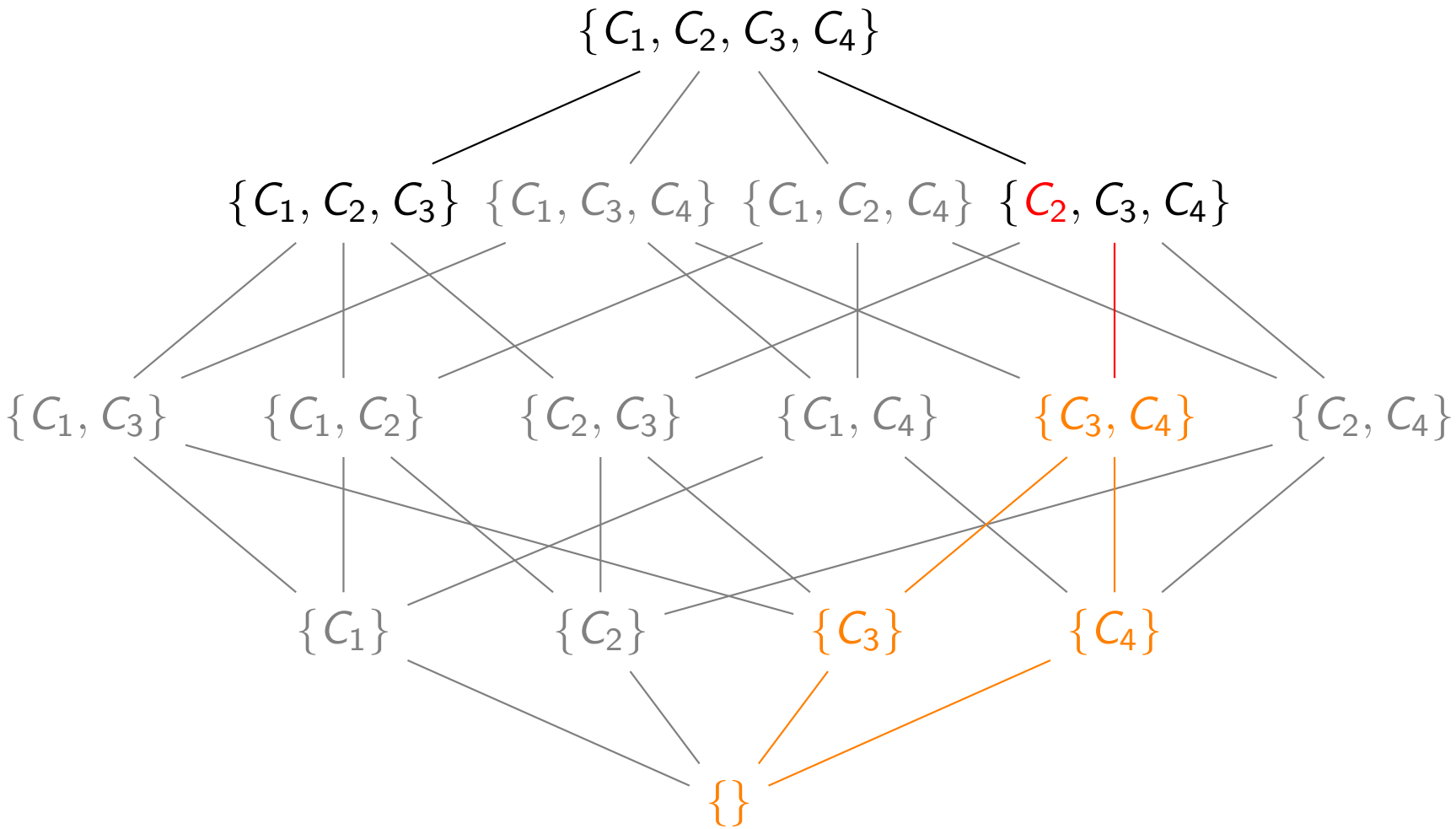
Definition

Ein **nach unten gerichteter Keil unerfüllbarer Untermengen (UNSAT-Keil)** besteht aus einer unerfüllbaren Untermenge $F' \subseteq F$ und allen ihren Obermengen, die dann ebenfalls unerfüllbar sind.

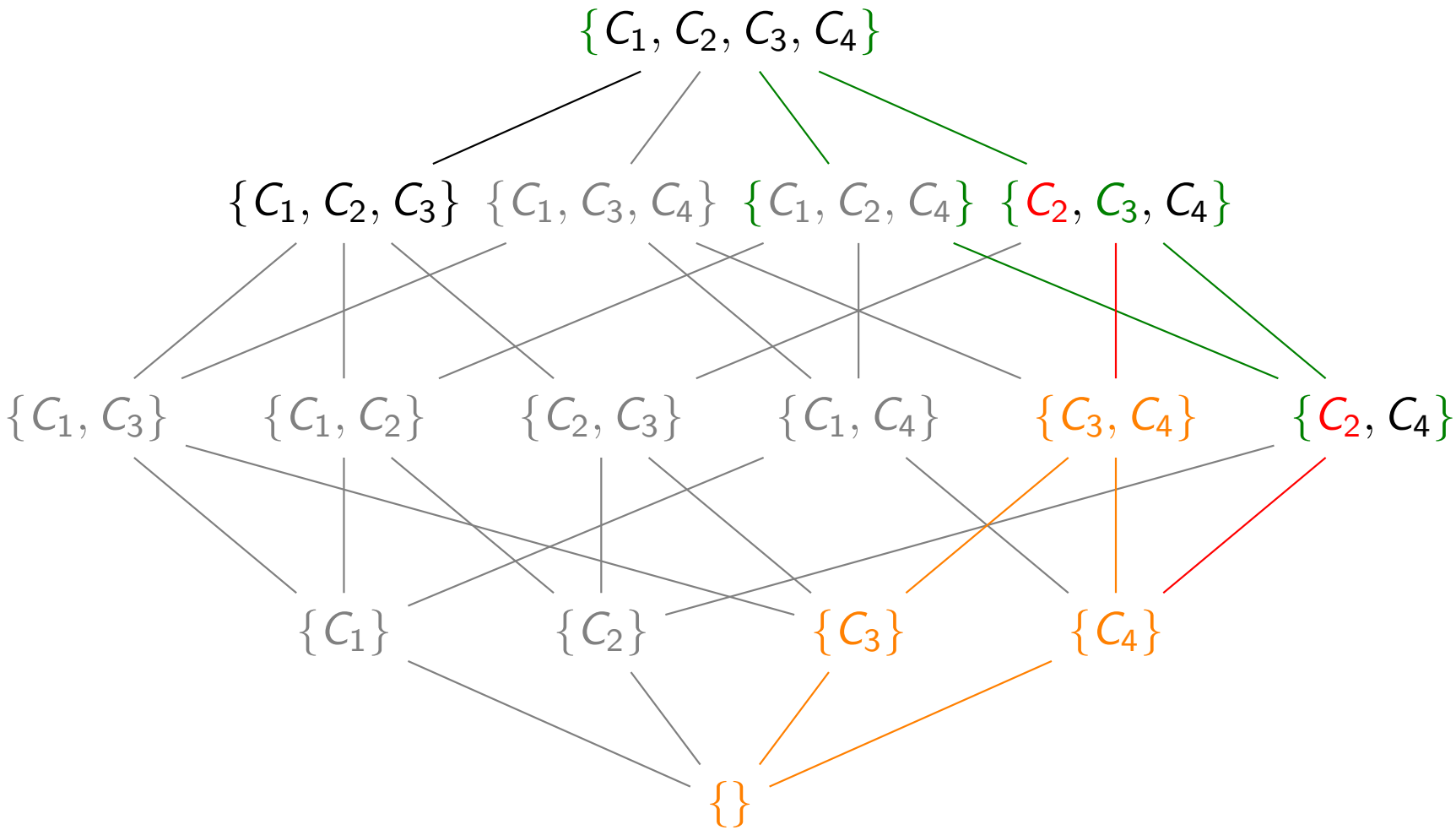
Repräsentation

- für $F \backslash (F' \backslash \{C\}) = \{C_1, ..., C_n\}$ wissen wir jetzt, dass $(s_1 \wedge ... \wedge s_n)$ eine hinreichende Bedingung für Unerfüllbarkeit ist
- das lässt sich nicht als Klausel schreiben
- entspricht der positiven Repräsentation des neuen UNSAT-Keils

Beispiel: Lernen eines UNSAT-Keils



Beispiel: Lernen eines UNSAT-Keils



Probleme mit UNSAT-Keilen

Problem: UNSAT-Keile sind Minterme

- wir können beide Arten von Wissen nicht in einer CNF-Instanz mischen (in kanonische Form zu bringen wäre viel zu aufwändig und könnte die Metainstanz exponentiell aufblähen)
- bräuchten ein zweites Meta-Problem in DNF, müssten dann darauf entsprechende Infrastruktur für schnelle UNSAT-Keil-Mitgliedschaftsüberprüfung entwickeln

UNSAT-Keile überflüssig?

- Klauselmengenverfeinerung begrenzt Nutzen von UNSAT-Keilen, die meisten redundanten Klauseln werden dadurch gefunden
- bei anderen MUS-Extraktionsalgorithmen sieht die Lage anders aus (bei Durchquerung des Teilmengenverbands von unten enthielten UNSAT-Keile die wichtige Information!)

Metainstanz: Subsumption Checking

Problem: SAT-Keile können einander enthalten

- mit mehr Wissen können sich SAT-Keile vergrößern, wenn wir feststellen, dass schon eine Obermenge einer bereits bekannten erfüllbaren Untermenge erfüllbar ist
- solche Relationen zu erkennen lohnt sich, denn nur der jeweils größere SAT-Keil muss gespeichert werden, und die Metainstanz kann sich dadurch stark vereinfachen

Lösung: Subsumption Checking

- die Teilmengenrelationen zwischen SAT-Keilen erkennt man durch einen **rückwärtsgerichteten Subsumption Check**: für jede neue Metaklausel wird geprüft, ob sie schon vorhandene Klausel subsumiert; subsumierte Klauseln werden entfernt
- der umgekehrte Fall kommt nicht vor

Metainstanz: Das Größenproblem

Betrachtung zur Länge der Metaklauseln

- die Zahl der Selektorvariablen in einer Metaklausel hängt an der Größe des repräsentierten SAT-Keils: je kleiner der SAT-Keil, über desto mehr Selektorvariablen muss die Disjunktion gebildet werden
- MUSes in der Regel eher klein im Verhältnis zur Größe von F $\Longrightarrow$ sehr lange Klauseln!

Betrachtung zur Zahl und Überlappung der Metaklauseln

- besonders in der Endphase der MUS-Extraktion hat man viele fehlgeschlagene Reduktionsversuche in der gleichen US
- man erhält Metaklauseln $\{s_{17}, \ldots, s_{1456}, s_3\}$, $\{s_{17}, \ldots, s_{1456}, s_5\}$, $\{s_{17}, \ldots, s_{1456}, s_{11}\}$, $\{s_{17}, \ldots, s_{1456}, s_{12}\}$, $\{s_{17}, \ldots, s_{1456}, s_{16}\}$ ...
- dort hilft auch Subsumption Checking nichts, man hat also sehr viele sehr lange Klauseln, die sich nur in einem Literal unterscheiden

Benchmark der eingesparten SAT-Aufrufe

Instance name	no sharing	Random initial reduction graph depth 0	1	2	3
C168_FW_SZ_66.cnf	$64 + 7$	$56 + 7$	$56 + 6$	$49 + 5$	$46 + 6$
C170_FR_SZ_96.cnf	$168 + 31$	$118 + 30$	$116 + 29$	$107 + 29$	$102 + 29$
C202_FS_SZ_97.cnf	$108 + 12$	$94 + 12$	$84 + 11$	$79 + 9$	$67 + 10$
C202_FW_SZ_103.cnf	$466 + 11$	$463 + 11$	$460 + 10$	$442 + 10$	$439 + 10$
C220_FV_SZ_55.cnf	$915 + 14$	$806 + 14$	$806 + 13$	$801 + 13$	$798 + 13$

Anmerkungen

- Ergebnis der Durchläufe dreier Reduktionsagenten mit unterschiedlichen Standardheuristiken
- um vorherige Kenntnis des Reduktionsgraphen zu simulieren, wurden in jedem Knoten zufällig fünf Knoten von unbekanntem Status gewählt und reduziert, und das rekursiv bis zu einer Tiefe n
- links die Zahl der SAT-Aufrufe mit Ergebnis SAT, rechts UNSAT

Übersicht

Blockpartition: Definitionen

Definition

Eine **Blockpartition** ist eine Menge $\mathcal{B} := \{B_1, \ldots, B_k\}$ von Klauselmengen oder **Blöcken** $B_j \subseteq F$ so, dass für alle $1 \leq i \neq j \leq k$ gilt $B_i \cap B_j = \emptyset$, d.h. die Blöcke sind disjunkt. Wir stellen zusätzlich die Bedingung $\bigcup B_j = F$, d.h. jede Klausel in F muss einem Block zugeordnet sein.

Definition

Die **Blockzugriffsfunktion** ist definiert als $block(s_i) := B_j$ mit $C_i \in B_j$.

Definition

Für einen ganzzahligen Schwellenwert $n_{max} \geq 0$ heißt eine Metainstanz $G := \{D_1, \ldots, D_n\}$ für eine CNF-Instanz F **$\mathbf{n}_{max}$-repräsentierbar** in einer Blockpartition $\mathcal{B}$ über F gdw. für jedes $1 \leq i \leq n$ und jedes $s_{ij} \in D_i$ mit $|block(s_{ij}) \cap D_i| > n_{max}$ gilt $block(s_{ij}) \subseteq D_i$.

Blockpartition: Eigenschaften

Kompression durch eine Blockpartition

Ist eine Metainstanz G für F n_{max}-repräsentierbar in einer Blockpartition $\mathcal{B}$ über F, so kann man sehr intuitiv eine **Kompression** G' von G durch $\mathcal{B}$ definieren, die eine neue Selektorvariable für jeden Block einführt.

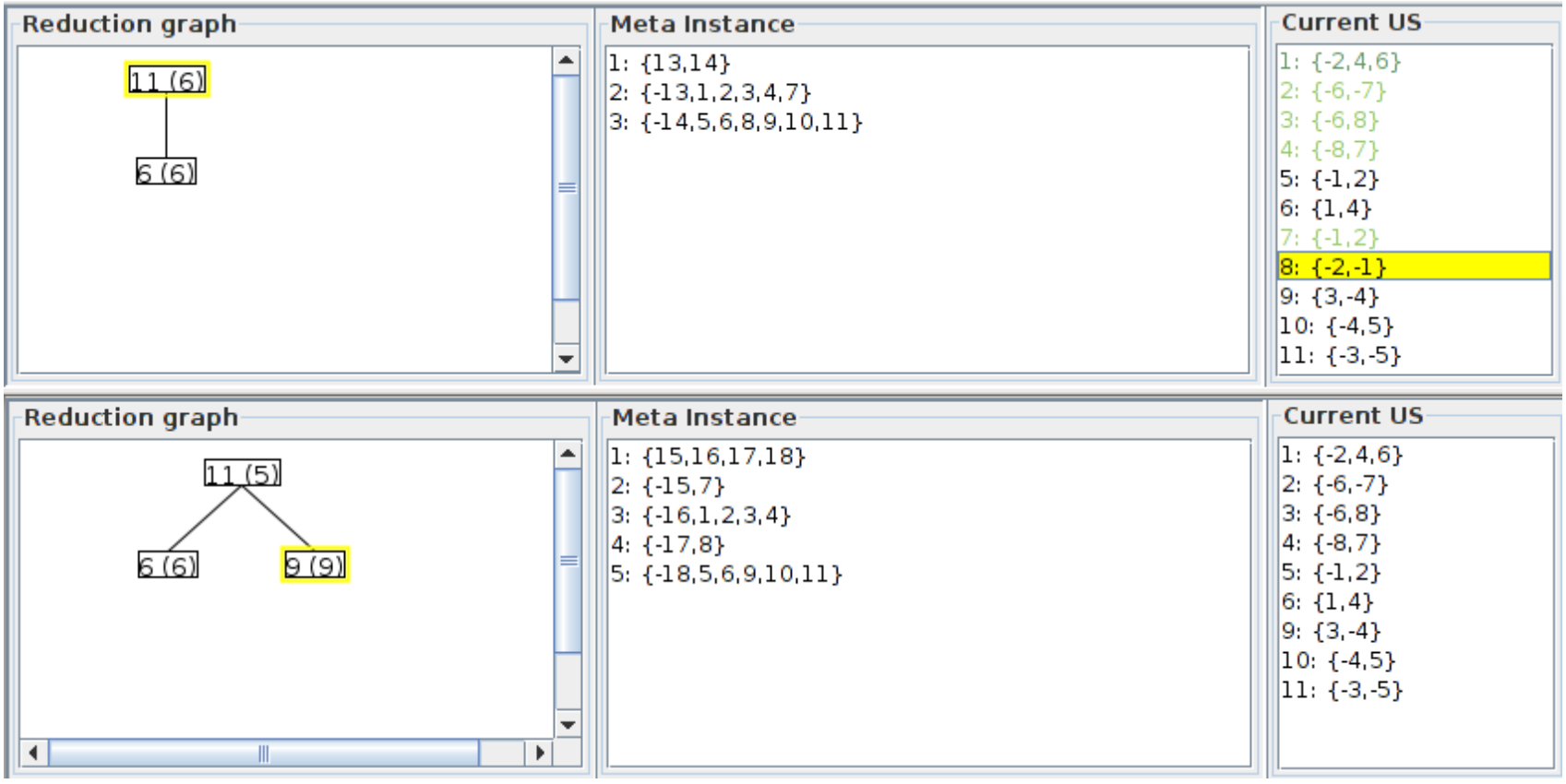
Erfüllbarkeitsäquivalenz der Kompression

Für eine CNF-Instanz F und eine Blockpartition $\mathcal{B}$ über F ist eine beliebige Metainstanz G, die in $\mathcal{B}$ n_{max}-repräsentierbar ist, erfüllbar gdw. die Kompression G' von G durch $\mathcal{B}$ erfüllbar ist.

Erhalt der Propagation unter Metainstanzkompression

Für die Repräsentation $\{l_1, \ldots, l_k\} := \{s_i | C_i \in F'\} \cup \{\neg s_i | C_i \notin F'\}$ eines $F' \subseteq F$ und jede Kompression G' einer Metainstanz G für F wird jede durch Propagation auf $G \cup \{\{l_1\}\} \cup \cdots \cup \{\{l_k\}\}$ hergeleitete Unitklausel auch durch Propagation auf $G' \cup \{\{l_1\}\} \cup \cdots \cup \{\{l_k\}\}$ hergeleitet.

Blockpartition: Beispiel und Visualisierung



Blockbaum: Definition

Definition

Ein **Blockbaum** ist ein Tupel $\mathcal{BT} := (V, E) := (\{B_0, B_1, \ldots, B_k\}, \{(B_i, B_j) \mid B_i \supset B_j,\ \neg\exists h\colon B_i \supset B_h \supset B_j\})$, wieder basierend auf Blöcken $B_j \subseteq F$. Wir fordern für alle Blöcke B_h, B_i, B_j, dass $(B_h, B_i), (B_h, B_j) \in E \Rightarrow B_i \cap B_j = \emptyset$, sowie $\bigcup_{(B_i,B_j)\in E} B_j = B_i$, womit jedes Element eines Blocks auch in genau einem Kindblock ist. Außerdem fordern wir die Existenz eines **Wurzelblocks** $F =: B_0 \in V$.

Definition

Menge der **Blattblöcke** $L(\mathcal{BT}) := \{B_i \in V \mid \neg\exists B_j \in V\colon (B_i, B_j) \in E\}$, **leafBlock($s_i$)** bezeichnet den zu C_i gehörenden Blattblock $B_k \in L(\mathcal{BT})$

Eigenschaften

- $L(\mathcal{BT})$ bildet eine Blockpartition über F
- n_{max}-Repräsentierbarkeit mittels $leafBlock(s_i)$ statt $block(s_i)$

Blockbaum: Eigenschaften

Kompression durch einen Blockbaum

Ist eine Metainstanz G für F n_{max}-repräsentierbar in einem Blockbaum $\mathcal{BT}$ über F, so kann man eine **Kompression** G^T von G durch $\mathcal{BT}$ definieren, die eine neue Selektorvariable für jeden Block einführt und die $\subset$-Verhältnisse zwischen Blöcken durch zusätzliche Metaklauseln kodiert.

Erfüllbarkeitsäquivalenz der baumbasierten Kompression

Für einen Blockbaum $\mathcal{BT}$ über F ist eine in $\mathcal{BT}$ n_{max}-repräsentierbare Metainstanz G erfüllbar gdw. es die Kompression G^T durch $\mathcal{BT}$ ist.

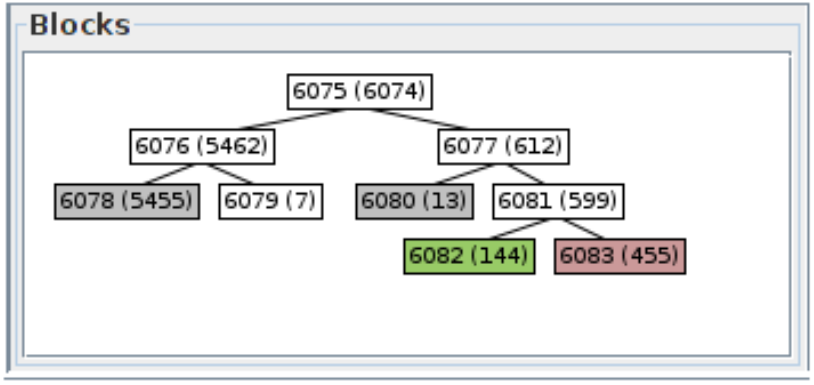
Erhalt der Propagation unter baumbasierter Kompression

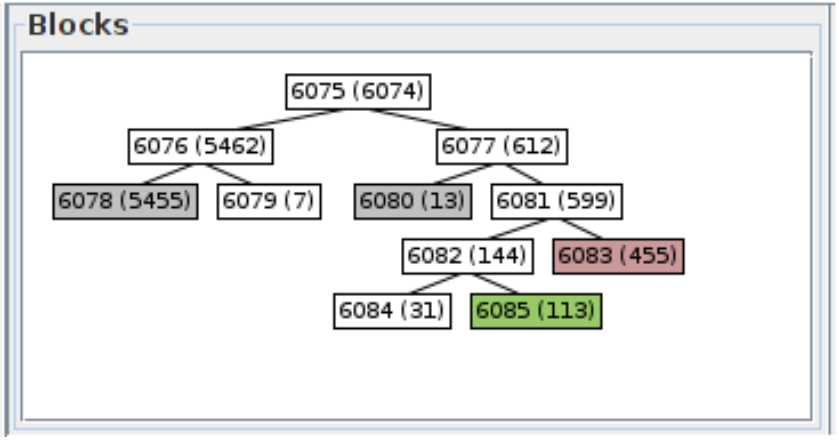
Für die Repräsentation $\{l_1, \ldots, l_k\} := \{s_i | C_i \in F'\} \cup \{\neg s_i | C_i \notin F'\}$ eines $F' \subseteq F$ und jede Kompression G^T einer Metainstanz G für F wird jede durch Propagation auf $G \cup \{\{l_1\}\} \cup \cdots \cup \{\{l_k\}\}$ hergeleitete Unitklausel auch durch Propagation auf $G' \cup \{\{l_1\}\} \cup \cdots \cup \{\{l_k\}\}$ hergeleitet.

Blockbaum: Beispiel und Visualisierung

Funktionalität der Visualisierung

- beim Lernen von Metaklauseln wird ein Blockbaum inferiert, indem G stets n_{max}-repräsentierbar gehalten wird
- beim Doppelklick auf einen Block im Baum wird zunächst versucht, alle Elemente des Blocks simultan zu löschen; misslingt dies, werden die Elemente einer nach dem anderen reduziert, bis der gesamte Block als kritisch bestimmt wurde oder sich spaltet





Übersicht

Zentrale Erkenntnisse aus der Diplomarbeit

Interaktive MUS-Extraktion

- die Suchräume bei der Suche nach MUSes können sehr komplexe Struktur haben, deren interaktive Erkundung lohnenswert erscheint
- aufbauend auf Minisat und ein existierendes Framework für grafisches Debuggen ließ sich die Idee in einem Prototypen umsetzen

Metalernen von SAT-Keilen

- durch Verwaltung einer Metainstanz über den Selektorvariablen kann man die zunehmende Kenntnis des Suchraums repräsentieren und dadurch SAT-Aufrufe einsparen
- aus Effizienzgründen muss die Metainstanz komprimiert werden, dabei entstehen wiederum interessante Strukturen, die Aufschluss über das Zusammenspiel von Klauselgruppen geben

Anwendung auf industrielle Instanzen

Problem: keine symbolische Information

- die Evaluation anhand von Anwendungsinstanzen ist schwierig, weil in Benchmarks normalerweise keine Information über die Semantik der Symbole enthalten ist (oft aus Geheimhaltungsgründen)
- selbst wenn Instanzen mit symbolischer Information beschaffbar gewesen wären, hätte die Evaluation vermutlich zu viel Einarbeitung in einen Anwendungskontext erfordert

Eigene Anwendung: Grammar Engineering

- in Kapitel 6 der Diplomarbeit Entwicklung einer SAT-Kodierung für ein Problem aus dem eigenen Anwendungsbereich: die Fehlersuche in symbolischen Grammatiken natürlicher Sprache
- gefundene MUSes gut interpretierbar, Suchräume haben interessante Struktur; Reduktionsgraphen sind aber generell zu flach und schöpfen Potential nicht aus, daher wenig aussagekräftig

Literaturhinweise

Diplomarbeit mit Beweisen und weiterführenden Gedanken

Johannes Dellert: **Interactive Extraction of Minimal Unsatisfiable Cores Enhanced By Meta Learning**. Uni Tübingen 2013.

Löschungsbasierte MUS-Extraktion mit Selektorvariablen

Alexander Nadel: **Boosting minimal unsatisfiable core extraction**. Formal Methods in Computer-Aided Design (FMCAD) 2010.

Neuere Entwicklungen in der MUS-Extraktion

João P. Marques-Silva & Inês Lynce: **On Improving MUS Extraction Algorithms**. Theory & Applications of Satisfiability Testing (SAT) 2011.

Schluss & Diskussion

Vielen Dank für die Aufmerksamkeit! Fragen?