

Verarbeitung Robuster Instruktionen

ATMS -
An Assumption-Based
Truth Maintenance System

Gliederung

1. Probleme der TMS
2. Grundzüge der ATMS
 1. Problem-solver Architektur
 2. Definitionen
 3. Labels
 4. Datenstrukturen

Gliederung(Fortsetzung)

2. Grundzüge der ATMS(Fortsetzung)
 5. Basis-Operationen
 6. Algorithmen - Label Update
 7. Sonstiges
3. Aufhebung der Probleme der TMS
4. Diskussion

1.

Probleme der TMS

1. Probleme der TMS

- the single-state problem.
- overzealous contradiction avoidance.
- switching states is difficult.
- the dominance of justification.
- the machinery is cumbersome.
- unouting

single-state problem

- Gegeben sei eine Menge von Annahmen, die mehrere Lösungen haben kann , z.B.:
 $x, y, z \in \{0, 1\}, x \neq y, y \neq z$
- TMS Algorithmus erlaubt es nur, maximal eine Lösung zu berücksichtigen
- schwierig, zwei gleichermaßen plausible Lösungen gleichzeitig zu überprüfen

overzealous contradiction avoidance

- TMS versucht "übereifrig" Widersprüche zu vermeiden
- TMS wird bei Widerspruch von *A und B* garantiert entweder *A* oder *B* behandeln, aber nicht beide
- keine gute Taktik, da eigentlich Schlüsse, die sowohl auf *A* als auch auf *B* begründet sind, vermieden werden müssten
- eigentlich wichtig weiterhin Schlüsse von *A* und *B* unabhängig voneinander ziehen zu können

switching states is difficult

- TMS hat keinen Begriff eines globalen Zustands
- es können keine temporären Änderungen vorgenommen werden, weil nicht zum vorherigen Zustand zurückgewechselt werden kann
- ungünstige Lösung: Snapshot machen und danach vom Snapshot aus resetten

the dominance of justifications

- der Begriff der "assumption" ist kontextabhängig
- "assumption" ist jeder Knoten dessen unterstützende Begründung von einem anderen Knoten abhängt, der in der out Liste ist
- während der Prozess des Problemlösens abläuft, können sich diese unterstützenden Begründungen ändern und somit ändert sich die Menge der Knoten, die als "assumption" angesehen wird
- problematisch für Problem Solver, die häufig auf "assumptions" und Begründungen zugreifen

the machinery is cumbersome

- einige Algorithmen der TMS sehr ressourcengefräßig und langandauernd (backtracking, Finden einer Lösung, die alle Begründungen erfüllt, ...)
- in der Zwischenzeit können sich Daten mehrmals geändert haben (von believed nach not believed und zurück)

unouting

- Szenario: Datum wird abgeleitet, durch einen Widerspruch verworfen und durch einen weiteren Widerspruch wieder gültig
- Problem: mit diesem Datum zusammenhängende Daten, die vormals abgeleitet wurden, müssen nun (eventuell) nochmal abgeleitet werden (ineffizient)
- Lücke zwischen Daten, die im alten Kontext nicht abgeleitet werden konnten, im neuen aber schon

Grund für die (meisten) Probleme der TMS

- Datum ist allein mit der Menge an Begründungen markiert
- Begründungen bestimmen, wie sich Datum von anderen Daten ableitet
- Kontext („in welchem Datum gilt,“) wird also nur implizit über Begründungen beschrieben
- Konsequenz: Datenbank muss konsistent gehalten werden, darf nur einen Kontext zur selben Zeit beschreiben
- unmöglich, direkt auf den Kontext zuzugreifen, in dem ein Datum gilt

Lösungsversuch

- Begründungen der TMS werden um ein Label erweitert
- Label beschreibt explizit die Zusammenhänge, in denen ein Datum gilt

2.

Die Grundzüge der ATMS

2. Grundzüge der ATMS

- ATMS ist effizienter als herkömmliche TMS
- Schnittstelle zwischen Problem Solver und TMS ist stimmiger
- ATMS kann im Gegensatz zur TMS mehr als einen Punkt im Suchraum zur gleichen Zeit untersuchen
- ATMS hat nicht die Probleme der TMS

2. Grundzüge der ATMS

- ATMS vermeidet dependency-directed backtracking der TMS
- ATMS berechnet Mengen von Annahmen ("assumptions") anhand der vom PS gelieferten Begründungen
- Aus den Annahmen werden alle anderen Daten abgeleitet
- Annahmenmengen sind leichter modifizierbar als die durch sie repräsentiere Daten
- Datenbank muss nicht wie bei der TMS zwangsläufig konsistent gehalten werden

Behandlung von Inkonsistenz

- TMS: Alle Aussagen von Widerspruch betroffen

$$\begin{array}{ll}
 a, & c, \\
 a \rightarrow b, & c \rightarrow d \\
 a \wedge c \rightarrow e, & b \wedge d \rightarrow \perp
 \end{array}$$

- ATMS: MX/X bedeutet: X wird geglaubt, bis das Gegenteil bewiesen ist

$$: MA/A, \quad : MC/C, \\
 A \rightarrow b, \quad C \rightarrow d, \quad A \wedge C \rightarrow e$$

Repräsentation:

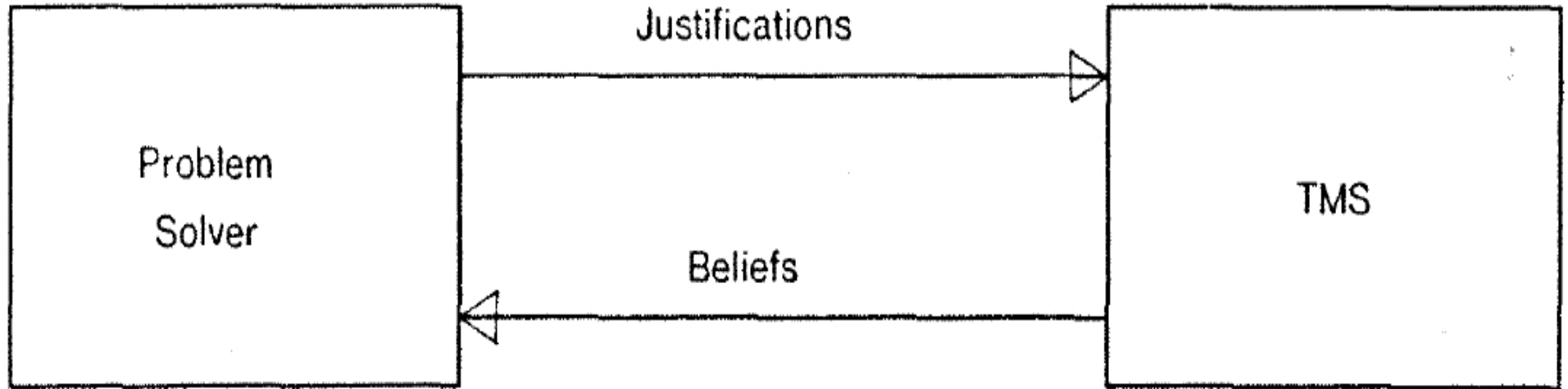
$$\langle b, \{\{A\}\}, \rangle, \quad \langle d, \{\{C\}\}, \rangle, \\
 \langle e, \{\{A, C\}\}, \rangle$$

Behandlung von Inkonsistenz

$: MA/A, \quad : MC/C,$ Nach Hinzufügen von $b \wedge d \rightarrow \perp$
 $: A \rightarrow b, \quad C \rightarrow d, \quad A \wedge C \rightarrow e$ wird $\langle e, \{\{A, C\}\}, \rangle$ zu
 $\langle e, \{\}, \rangle$

- ATMS: Widerspruch berührt allein e
- A und C dürfen nicht zusammen gelten
- b und d sind nicht vom Widerspruch berührt
- Auflösung des Widerspruchs durch Löschen der inkonsistenten Annahmemenge $\{A, C\}$

2.1 Problem-solver Architektur



$P(a)$
 $\forall x P(x) \rightarrow Q(x)$
 $Q(a)$

$\gamma_{P(a)}, \gamma_{\forall x P(x) \rightarrow Q(x)} \Rightarrow \gamma_{Q(a)}$

PS kennt Schlussregeln, Schlüsse werden an TMS weitergeleitet
TMS benutzt Begründungen als Aussagen und schliesst daraus die Beliefs

2.1 Problem-solver Architektur

- Ausdruck hat für den PS interne Struktur
- der selbe Ausdruck hat für TMS atomare Struktur, behandelt Ausdruck als Aussage
- TMS bestimmt anhand des Ausdrucks und der Begründungen, welches Symbol folgt
- zwei verschiedene Schlussverfahren, die aber auf demselben Ausdruck arbeiten
- Problemlösen: Solange Begründungen ansammeln und Beliefs ändern bis Ziel erreicht ist

2.2 Definitionen

- Annahme ("assumption"): bezeichnet die Entscheidung etwas anzunehmen
- Angenommenes Datum ("assumed datum"): Datum des PS, von dem angenommen wird, dass es gilt
- Knoten("node"): entspricht einem PS Datum, eine Annahme ist ein spezieller Knoten

2.2 Definitionen

- Begründungen("justifications"): bestimmen, wie ein Knoten von anderen Knoten abgeleitet werden kann
Begründungen sind Implikationen: $x_1 \wedge x_2 \wedge \dots \rightarrow n$
- Umgebung("environment"): Menge von Annahmen (Konjunktion von Annahmen)
Knoten n gilt in Umgebung E , wenn n von E anhand der Menge an Begründungen J abgeleitet werden kann.
- Kontext("context"): Menge, gebildet aus den Annahmen einer konsistenten Umgebung und aller von diesen aus ableitbaren Knoten

2.2 Definitionen

- Charakterisierende Umgebung (für einen Kontext): Menge von Annahmen, aus der jeder Knoten des Kontextes ableitbar ist.

Ziel der ATMS

- ATMS bekommt Menge von Annahmen und Begründungen geliefert
- ATMS soll effizient die Kontexte bestimmen
- Für den PS wichtig bei Kontexten: konsistent oder nicht, weniger wichtig: zu wissen, welche Knoten im Kontext enthalten sind

2.3 Labels

- Label: Menge von Umgebungen, die jedem Knoten zugeordnet ist
- Jede Umgebung E im Label ist konsistent
- Label beschreibt die Annahmen, von denen der Knoten letztendlich abhängt
- Label wird von der ATMS konstruiert
- Begründung beschreibt, wie Knoten vom Vorgängerknoten abhängt, Label beschreibt wie Knoten letztendlich von Annahmen abhängt

2.3 Labels

- Label für Knoten n ist
 - konsistent, wenn alle Umgebungen E konsistent sind
 - korrekt, wenn sich n aus jeder Umgebung ableiten lässt
 - vollständig, wenn jede konsistente Umgebung E Obermenge einer Umgebung E' des Labels ist (Umgebungen werden wie Mengen behandelt)
 - minimal, wenn keine Umgebung des Labels eine Obermenge einer anderen ist

2.3 Labels

- n ist ableitbar, genau dann wenn E Obermenge einer anderen Umgebung des Labels ist
- Anhand des Labels kann direkt bestimmt werden, ob n gilt oder nicht
 - Label leer: n ist nicht ableitbar
 - Label nicht leer: n ist von einer konsistenten Menge an Annahmen ableitbar

2.3 Labels

- n ist im Kontext, wenn n aus den Annahmen der Umgebung, welche den Kontext charakterisiert, ableitbar ist.
Leicht erkennbar, da
 - wenn Knoten mindestens eine Umgebung hat, die Untermenge einer Umgebung ist, die den Kontext charakterisiert, dann ist Knoten im Kontext
 - ob ein Knoten in einem Kontext ist, kann effizient mit einem Untermengentest geprüft werden
- bei konventioneller TMS: Kontext nur implizit über Begründungen erschliessbar

2.4 Datenstrukturen

- Knoten: $\gamma_{\text{datum}} : \langle \text{datum}, \text{label}, \text{justifications} \rangle$
 - Datum wird vom PS geliefert (wird vom ATMS nicht untersucht)
 - Begründungen werden vom PS für das Datum erstellt (werden vom ATMS untersucht, aber nicht verändert)
 - Label wird vom ATMS berechnet (kann nicht vom PS geändert werden)

2.4 Datenstrukturen

$\langle p, \{\{\}\}, \{(\)\} \rangle$ Fakt

$\langle A, \{\{A\}\}, \{(A)\} \rangle$ Annahme

$\langle a, \{\{A\}\}, \{(A)\} \rangle$ angenommener Knoten

$\gamma_{\perp} : \langle \perp, \{\}, \{\dots\} \rangle$ *falsity*

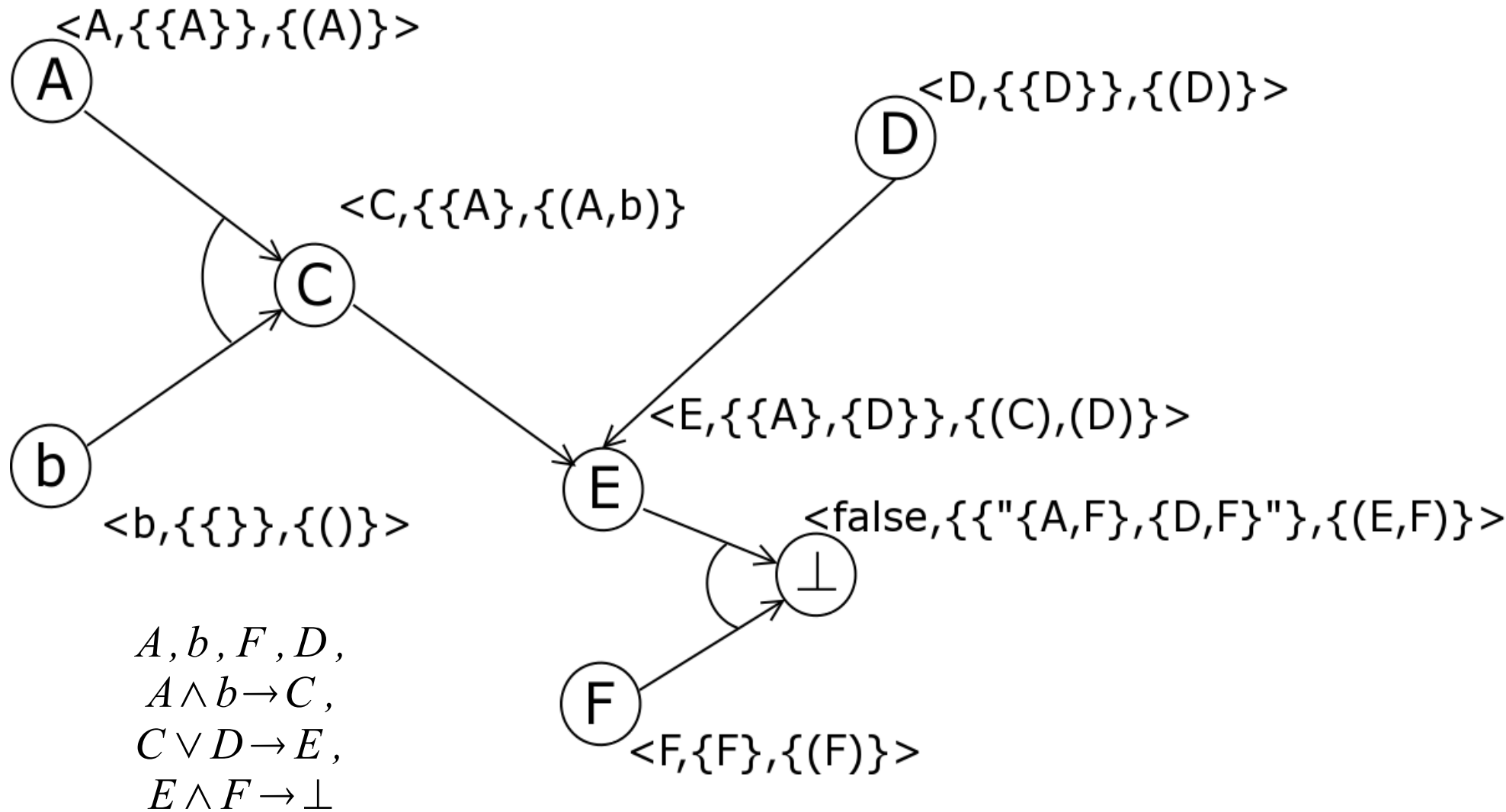
Allgemein:

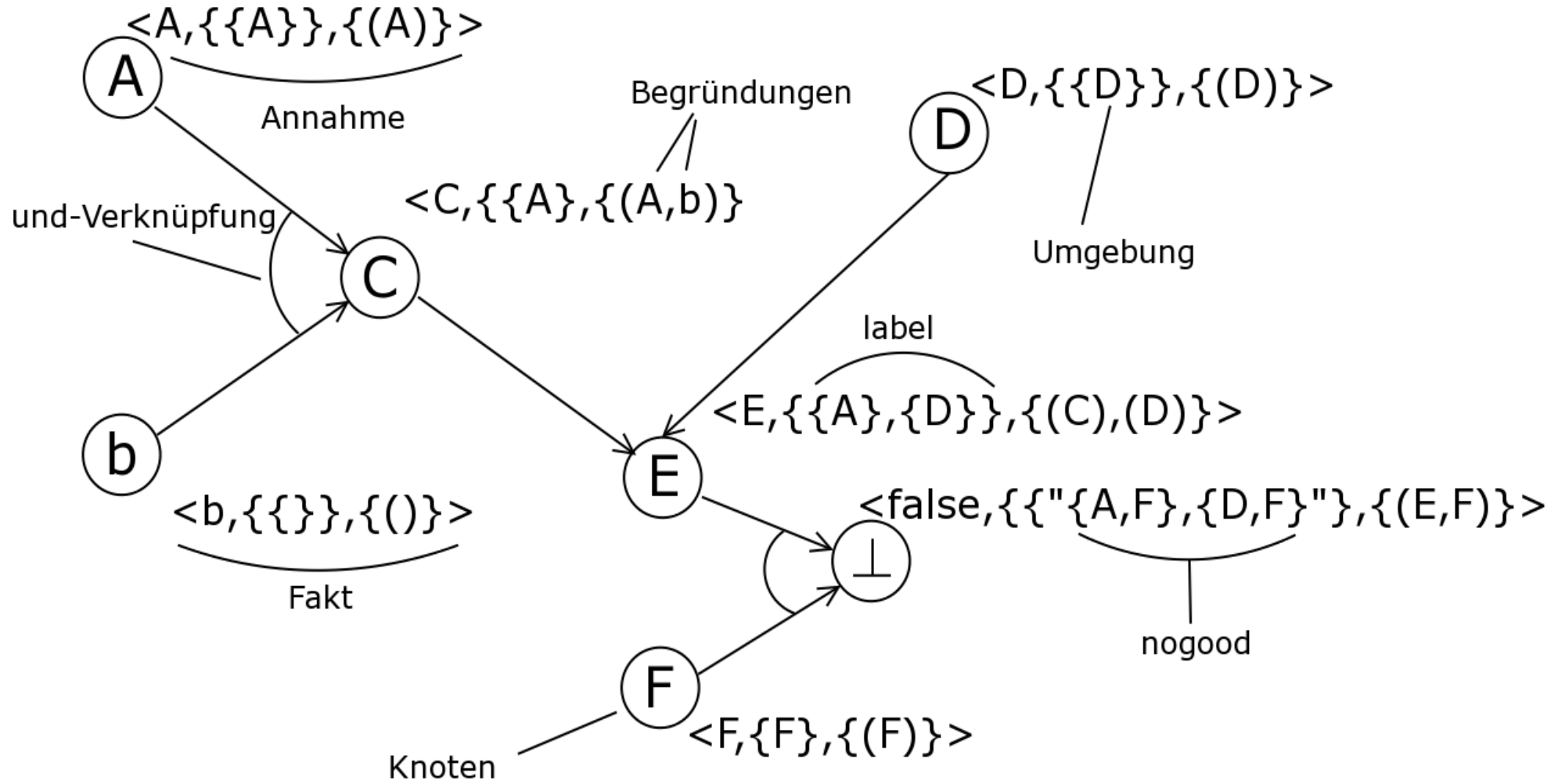
$\langle n, \{\{A_1, A_2, \dots\}, \{B_1, B_2, \dots\}, \dots\}, \{(z_1, z_2, \dots)(y_1, y_2, \dots)\dots\} \rangle$

repräsentiert die Implikationen:

$$(A_1 \wedge A_2 \wedge \dots) \vee (B_1 \wedge B_2 \wedge \dots) \vee \dots \rightarrow n$$

$$(z_1 \wedge z_2 \wedge \dots) \vee (y_1 \wedge y_2 \wedge \dots) \vee \dots \rightarrow n$$





2.5 Basis-Operationen

- 3 Basis-Operationen , alle anderen Operationen aufbaubar
- ATMS Knoten für ein Datum des PS erstellen
- Annahme erzeugen
- Begründung zu einem Knoten hinzufügen
- Algorithmen stellen sicher, dass nach Beendigung jeder Operation jedes Label jedes Knoten konsistent, korrekt, vollständig und minimal ist

2.6 Algorithmen - Label Update

I $\mathcal{Y}_{x+y=1} : \langle x+y=1, \{\{A, B, \}, \{B, C, D\}\}, \{\dots\} \rangle$

II $\mathcal{Y}_{x=1} : \langle x=1, \{\{A, C\}, \{D, E\}\}, \{\dots\} \rangle$

III $\mathcal{Y}_{y=0} : \langle y=0, \{\}, \{\} \rangle$

IV *nogood* $\{A, B, E\}$

PS schliesst $y=0$ aus $x+y=1$ und $x=1$ und übermittelt dem TMS folgende Begründung:

$$\mathcal{Y}_{x+y=1}, \mathcal{Y}_{x=1} \Rightarrow \mathcal{Y}_{y=0}$$

ATMS erneuert das Label von $\mathcal{Y}_{y=0}$ (Label ist korrekt, vollständig, konsistent, minimal)

korrekte und vollständige Vereinigungsmenge (aus I und II):

$\{\{A, B, C\}, \{A, B, C, D\}, \{A, B, D, E\}, \{B, C, D, E\}\}$

$\mathcal{Y}_{y=0} = \langle y=0, \{\{A, B, C\}, \{B, C, D, E\}\}, \{(\mathcal{Y}_{x+y=1}, \mathcal{Y}_{x=1})\} \rangle$

2.6 Algorithmen - Label Update

1. PS liefert dem ATMS eine neue Begründung
2. ATMS berechnet neues Label für den Knoten, wenn jeder Vorgängerknoten ein korrektes und vollständiges Label hat
3. Wenn das neuberechnete Label dasselbe wie das alte ist, dann ist der Knoten abgearbeitet

2.6 Algorithmen - Label Update

4. Wenn der neuberechnete Knoten \perp ist, dann wird jede Umgebung E des Labels zur *nogood* Datenbank hinzugefügt und inkonsistente Umgebungen werden aus dem Label jedes Knoten gelöscht
5. Wenn der neuberechnete Knoten nicht \perp ist, wird der Algorithmus auf die Nachfolgerknoten angewandt (Knoten mit Begründungen, die den geänderten Knoten beinhalten)
6. Prozess muss (irgendwann) terminieren, da es nur eine endliche Anzahl Annahmen gibt

2.7 Sonstiges

- effizient auch für viele Knoten
- Begründungen können zurückgenommen werden
- Klassen können eingeführt werden (Gruppierung von Knoten in Mengen)
- Manipulation von Annahmemengen kann effizient ausgeführt werden (durch Bitvektoren)

3.

Aufhebung der Grenzen der TMS

3 Aufhebung der Probleme der TMS

- single state problem: verschiedene, sich gegenseitig ausschliessende Lösungen können gleichzeitig (aber in verschiedenen Kontexten) auftreten
- overzealous contradiction avoidance: tritt ein Widerspruch zwischen A und B auf, so werden nur Daten entfernt, die sich auf A und B gleichermaßen begründen.
- switching states is difficult: trivial oder irrelevant, da Zustand durch Menge an Annahmen definiert ist

3 Aufhebung der Probleme der TMS

- dominance of justifications: Begründungen sind nicht vorherrschend, sondern Annahmen.
Annahmemengen können problemlos verarbeitet werden
- machinery is cumbersome: kein backtracking mehr erforderlich, Annahmen, die zu einem Widerspruch führen sind ablesbar
- unouting: ATMS "überlistet" unouting-Problem, ATMS überprüft alle Lösungen auf einmal (im Gegensatz zur TMS) und es gibt kein Backtracking oder Zurücknahme

4.

Diskussion