Constraintsystem

Definition 1 Ein *Constraintsystem* ist ein Tripel $(\Sigma, CT, \mathcal{C})$. Dabei sind

- Σ eine Signatur, wobei
  - $true \in \Sigma$ und $false \in \Sigma$,
- die *Constrainttheorie* CT eine nichtleere, konsistente Theorie über Σ, und
- die *erlaubten Constraints* $\mathcal{C}$ eine Menge von Formeln über Σ, wobei
  - $true \in \mathcal{C}$ und $false \in \mathcal{C}$
  - $\mathcal{C}$ abgeschlossen ist unter Konjunktion und Variablenumbenennung (Variantenbildung) $\lhd$

Mai 1998

Anmerkung

Im Höhefeld-Smolka-Schema wird in einem Constraintsystem angegeben, um welche Constraintsymbole es sich handelt, wie die Constraintsymbole definiert sind, und welche Constraints erlaubt sind.

Eine allgemeinere Bedingung ist, Abschluß unter existenzieller Quantifizierung zu fordern. Für unsere Zwecke reicht die angegebene Bedingung aus.

Die Definition der erlaubten Constraints $\mathcal{C}$ stellt sicher, daß wir mit Konjunktionen von Constraints arbeiten können, d. h., daß wir Constraints als Ziele in den Rümpfen von Klauseln verwenden können. Gleichzeitig schränkt $\mathcal{C}$ die Formeln so ein, daß eine effiziente Implementierung des zugehörigen Constraintlösers möglich ist.

Beispiel Constraintsystem E

Das Constraintsystem E (engl. Herbrand system) stellt Constraints für die Gleichheit von Termen, d.h. endlichen Bäumen (engl. finite trees), zur Verfügung.

Signatur von E enthält:
Unendlich viele Funktionssymbole (mind. eine Konstante)
Constraintsymbole $\doteq$, *true* und *false*.

Constrainttheorie von E:
Clark'sche Gleichheitstheorie CET.

Erlaubte Constraints von E:

$$C \quad ::= \quad s \doteq t \quad | \quad true \quad | \quad false \quad | \quad C \wedge C$$

wobei s und t Terme über der Signatur von E sind.

Anmerkung

Im folgenden gehen wir davon aus, daß die Signatur Σ auch das Symbol $\doteq/2$ enthält. Dementsprechend fügt man die zugehörige Constrainttheorie CET zu CT und die Gleichheit zwischen bestimmten Termen als erlaubtes Constraint zu $\mathcal{C}$ hinzu.

Clarksche Gleichheitstheorie CET

Definition 2 Sei Σ eine Signatur, die unendlich viele Funktionssymbole, davon mindestens eine Konstante, enthält. Die *Clarksche Gleichheitstheorie CET* ist:

Reflexivität:
$\forall(true \rightarrow x \doteq x)$
Symmetrie:
$\forall(x \doteq y \rightarrow y \doteq x)$
Transitivität:
$\forall(x \doteq y \wedge y \doteq z \rightarrow x \doteq z)$
Verträglichkeit:
$\forall(x_1 \doteq y_1 \wedge \ldots \wedge x_n \doteq y_n \rightarrow f(x_1, \ldots, x_n) \doteq f(y_1, \ldots, y_n))$
Zerlegung:
$\forall(f(x_1, \ldots, x_n) \doteq f(y_1, \ldots, y_n) \rightarrow x_1 \doteq y_1 \wedge \ldots \wedge x_n \doteq y_n)$
Widerspruch:
$\forall(f(x_1, \ldots, x_n) \doteq g(y_1, \ldots, y_m) \rightarrow false)$ falls $f \neq g$ oder $n \neq m$
Azyklizität:
$\forall(x \doteq t \rightarrow false)$ falls t keine Variable ist und x in t vorkommt

Anmerkung

Damit das Gleichheitszeichen $\doteq$ richtig behandelt wird und nicht wie ein beliebiges binäres Prädikat, werden die Sätze, die die Gleichheit beschreiben (Clark'sche Gleichheitstheorie), zu $P^{\leftrightarrow}$ hinzugenommen.

Die ersten drei Sätze sagen aus, daß $\doteq$ eine Äquivalenzrelation ist.

Erwünschte Eigenschaften

Definition 3 Eine Theorie $\mathcal{T}$ heißt *vollständig* (engl. complete), falls für jede abgeschlossene Formel φ gilt: $\mathcal{T} \models \varphi$ oder $\mathcal{T} \models \neg\varphi$. $\lhd$

Definition 4 Ein Constraintsystem $(\Sigma, CT, \mathcal{C})$ heißt *erfüllbarkeitsvollständig* genau dann, wenn entweder $CT \models \exists C$ oder $CT \models \neg\exists C$ für alle Constraints $C \in \mathcal{C}$ gilt. $\lhd$

Anmerkung

Als Constrainttheorie CT hat man gerne eine *vollständige Theorie*. Es reicht aber aus, wenn CT eine schwächere Vollständigkeitsbedingung erfüllt, da wir es in unseren Kalkülen nur mit existenziell quantifizierten Variablen zu tun haben. Der Unterschied zur Vollständigkeit liegt in der Einschränkung auf existenzquantifizierte Formeln und nur solche aus $\mathcal{C}$.

Beispiel Vollständigkeit

Die Gleichheitstheorie CET des Constraintsystems E ist vollständig. Wenn die Signatur von E nur noch endlich viele Funktionssymbole (davon mindestens eine Konstante) enthielte, dann wäre CET nicht mehr vollständig, aber erfüllbarkeitsvollständig.

Anmerkung

Unabhängigkeit

Definition 5 Ein Constraintsystem $(\Sigma, CT, \mathcal{C})$ heißt *unabhängig* genau dann, wenn für alle Constraints $C, C_1, \ldots, C_n \in \mathcal{C}$ gilt:

$$CT \models \exists C \wedge \neg C_1 \wedge \ldots \wedge \neg C_n \text{ gdw } CT \models \exists C \wedge \neg C_i$$

$$\text{für alle } i = 1, \ldots, n.$$

$\lhd$

Anmerkung

Eine weitere wichtige Eigenschaft von Constraintsystemen ist die *Unabhängigkeit*. Sie erlaubt es, allgemeine Formeln über Constraints einfacher zu behandeln (siehe auch Abschnitt Constrainterweiterungen).

Die Unabhängigkeit besagt also, daß in einem unabhängigen Constraintsystem eine Disjunktion genau dann gültig ist, wenn zumindest eins der Disjunkte gültig ist bzw. eine Konjunktion mit negierten Constraints genau dann erfüllbar, wenn jedes negierte Constraint für sich mit allen positiven Constraints erfüllbar ist.

Beispiel Unabhängigkeit

Das Constraintsystem E mit zwei Funktionssymbolen (eine Konstante und einstelliges Funktionssymbol) ist nicht unabhängig:

$CET \models \exists X, Y, Z\ (X \doteq f(Y) \wedge \neg Y \doteq a \wedge \neg Y \doteq f(Z))$
$CET \models \exists X, Y\ (X \doteq f(Y) \wedge \neg Y \doteq a)$ und
$CET \models \exists X, Y, Z\ (X \doteq f(Y) \wedge \neg Y \doteq f(Z))$.

Anmerkung

Beispiel Unabhängigkeit

Ein Constraintsystem, das eine totale Ordnungsrelation $<$ für Zahlen zu Verfügung stellt, ist nicht unabhängig, da z. B.

$CT \models \exists X, Y\ \neg(X < Y)$,
$CT \models \exists X, Y\ \neg(X \doteq Y)$ und
$CT \models \exists X, Y\ \neg(Y < X)$ aber
$CT \not\models \exists X, Y\ (\neg(X < Y) \wedge \neg(X \doteq Y) \wedge \neg(Y < X))$.

Anmerkung

Lineare Gleichungssysteme bilden ein unabhängiges Constraintsystem.

Constraintlöser

Berechnungsdienste (engl. reasoning services):

- Konsistenztest (Erfüllbarkeitstest) (engl. satisfiability, consistency): Ist C in der Constrainttheorie erfüllbar, $CT \models \exists C$?
  Beispiele: $X > 3$, $X > Y$, $X > X$.

- Simplifikation (Vereinfachung): Wenn $CT \models \forall(C \leftrightarrow D)$ und D „einfacher“ ist als C, dann ersetze C durch D. Meistens wird D eine Normalform von C sein.
  Beispiele: $2 * X = 6$, $X^2 = X$, $X^2 > X$.

- Determination: Man will erkennen, wenn eine Variable nur noch einen einzigen Wert annehmen kann.
  Beispiel: $X \leq 2 \wedge 2 \leq X$.

Anmerkung

Ein *Constraintlöser* implementiert die Constraintheorie CT und liefert ein deterministisches Entscheidungsverfahren für ein Constraintsystem. Ein Constraintlöser sammelt die aus dem Programmablauf resultierenden Constraints, prüft ihre Erfüllbarkeit und vereinfacht und löst die Constraints.

Simplifikation:
Wenn ein Constraintlöser ein neu ankommendes Constraint zusammen mit den bisherigen Constraints vereinfachen kann, ohne die Vereinfachung der bisherigen Constraints zu wiederholen, sprechen wir davon, daß der Constraintlöser inkrementell ist. *Inkrementalität* wichtig für effiziente CLP-Sprachen. Eine Normalform erleichtert es, inkrementell zu sein.

Determination:
Damit koexistierende Constraintlöser Informationen über gemeinsame Variable austauschen können. Auch nützlich für die informative Darstellung einer Antwort.

Constraintlöser

Berechnungsdienste, Fortsetzung:

- Projektion: Man will existenziell quantifizierte Variablen eliminieren.
  Beispiel: $\exists Y\ (X \leq Y \wedge Y \leq Z)$.
- Folgerungstest (engl. entailment): Er ist erfolgreich ist, wenn $CT \models \forall(C \rightarrow D)$ gilt, er scheitert, wenn $CT \models \forall(C \rightarrow \neg D)$ gilt, ansonsten wird der Test *verzögert* (engl. delayed, suspended), bis er entschieden werden kann.
  Beispiele: $X > Y \rightarrow X \geq Y$,
  $X = Y \rightarrow X \leq Y$, $X \geq Y \rightarrow X > Y$.
- Negation: Implizit durch eigene Constraintsymbole (z. B. $=$ und $\neq$, $<$ und $\geq$), seltener durch expliziten Operator ($\neg$).

Anmerkung

Projektion:
Für Darstellung von Antworten wichtig und um eine Normalform klein zu halten. Aber je nach Constraintsystem und Algorithmus nicht immer (effizient) möglich. Z. B. in $X \doteq f(Y)$ die Variable Y nicht entfernbar.

Folgerungstest:
Der Folgerungstest ist die Grundlage für die Implementierung bedingter Anweisungen und nebenläufiger constraintbasierter Programmiersprachen.

Negation:
Die Negation ist i. A. nur effizient behandelbar, wenn das Constraintsystem unabhängig ist, weil dann jedes negierte Constraint für sich entschieden werden kann.

Alle angeführten Berechnungsdienste lassen sich als Varianten bzw. Erweiterungen des Vereinfachens mit Normalformen implementieren.

Selten stellt ein Constraintlöser alle Berechnungsdienste in vollem Umfang zur Verfügung, weil die Implementierung sonst zu komplex oder ineffizient würde.

Erwünschte Eigenschaften

Definition 6 Ein Constraintlöser ist

korrekt: Wenn $loese(C) = D$, dann $CT \models C \leftrightarrow D$.

vollständig: Wenn $CT \models \neg C$, dann $loese(C) = false$.

monoton: Wenn $loese(C) = false$, dann $loese(C \wedge D) = false$.

unabhängig von Variablennamen: Wenn C Variante von D, dann $loese(C) = loese(D)$.

inkrementell: Wenn Aufwand für $loese(loese(C) \wedge D)$ nicht größer als Aufwand für $loese(C \wedge D)$.

Anmerkung

Anfänge der Constraintbasierten Programmierung

1963	Sutherland, Sketchpad, Graphiksystem für geometrisches Zeichnen
1970	U. Montanari, Constraintnetzwerke
1970	R.E. Fikes, REF-ARF, Sprache für ganzzahlige, lineare Gleichungen
1972	A. Colmerauer, Marseille, sowie R.A. Kowalski, London, Prolog
1977	A.K. Mackworth, Constraints Netzwerk Algorithmen
1978	J.-L. Lauriere, Alice, Sprache für kombinatorische Probleme
1979	A. Borning, Thinglab, interaktives Graphiksystem
1980	G. L. Steele, Constraints, erste constraintbasierte Sprache, in LISP

Anmerkung

1982	A. Colmerauer, Prolog II, Prolog mit Gleichheitsconstraints
1987	Ait-Kaci, Austin, Life, Prologerweiterung mit Gleichheitsconstraints
1987	J. Jaffar und J.L. Lassez, CLP(X) - Schema
1987	J. Jaffar, CLP(R), Monash Univ. Melbourne, arithmetische Constraints
1988	P. v. Hentenryck, CHIP, ECRC München, endlichen Wertebereichen
1988	P. Voda, Vancouver, Trilogy, CLP-ähnlich mit Ganzzahlarithmetik
1988	Older, Ottawa, Bell-Northern Research, Intervallarithmetik
1988	A. Aiba, Tokyo, ICOT, nichtlineare Gleichungssysteme
1988	W. Leler, Termersetzungs-Sprache zum Schreiben von Constraints
1988	A. Colmerauer, Prolog III, Univ. Marseille, Constraints über Listen

Anmerkung

CLP-Sprachen vereinigen die Vorteile von LP-Sprachen (deklarativ, für beliebige Prädikate, nichtdeterministisch) mit denen von Constraintlösern (deklarativ, effizient für spezielle Prädikate, deterministisch). Die Kombination von Suche mit Constraintbehandlung ist besonders nützlich, wenn man kombinatorische Probleme (wie das vorgestellte Einfärbeproblem) lösen möchte, die meist exponentielle Komplexität haben, d. h. sehr schnell praktisch unlösbar werden. Die erhöhte Deklarativität verringert zudem den Programmieraufwand ohne die Effizienz zu vernachlässigen.

Atome:	A, B	::=	$p(t_1, \ldots, t_n),\ n \geq 0$
Constraints:	C, D	::=	$c(t_1, \ldots, t_n) \mid C \wedge D,\ n \geq 0$
Ziele:	G, H	::=	$\top \mid \bot \mid A \mid C \mid G \wedge H$
CL-Klauseln:	K	::=	$A \leftarrow G$
CL-Programme:	P, Q	::=	$\{K_1, \ldots, K_m\},\ m \geq 0$

Abbildung 1: CLP-Syntax

Anmerkung

Die CLP-Syntax (Abb. 1) unterscheidet sich von der LP-Syntax nur darin, daß Constraints eingeführt wurden, die in Zielen und daher im Rumpf einer Klausel auftreten können. In Abb. 1 sind die syntaktischen Ausdrücke, die gegenüber dem LP-Kalkül anders oder neu definiert sind, kursiv geschrieben.

CLP-Kalkül

Definition 7 Ein *Atom* ist ein Ausdruck der Form $p(t_1, \ldots, t_n)$, wobei p ein n-stelliges Symbol aus $\mathcal{P}_p$ und $t_1, \ldots, t_n$ Terme sind.

Ein *Constraintatom* ist ein Ausdruck der Form $c(t_1, \ldots, t_n)$, wobei c ein n-stelliges Symbol aus $\mathcal{P}_c$ ist und $t_1, \ldots, t_n$ Terme sind.

Ein *Constraint* ist entweder ein Constraintatom oder eine Konjunktion von Constraints.

Ein *Zielatom* ist entweder ein Atom oder ein Constraintatom.

Ein *Ziel* ist entweder $\top$ oder $\bot$ oder ein Zielatom oder eine Konjunktion von Zielen. $\lhd$

Anmerkung

Wir geben in Hinkunft bei den Kalkülen nur die essentiellen Definition explizit an, alle anderen Definitionen sind der Syntaxtabelle zu entnehmen bzw. werden von den vorhergehenden Kalkülen übernommen.

CLP-Kalkül

Definition 8 Ein *Zustand* ist ein Paar $\langle G, C\rangle$, wobei G ein Ziel ist und C ein Constraint. G wird Zielspeicher genannt und C Constraintspeicher.

Ein *Anfangszustand* ist ein Zustand von der Form $\langle G, \mathit{true}\rangle$.

Ein Zustand heißt *erfolgreicher Endzustand*, wenn er von der Form $\langle \top, C\rangle$ ist und C ungleich *false* ist.

Ein Zustand heißt *erfolgloser Endzustand* (gescheitert), wenn er die Form $\langle G, \mathit{false}\rangle$ hat.

Die Definitionen für die Arten von Ableitungen und Ziele aus dem LP-Kalkül gelten weiterhin. $\lhd$

Anmerkung

Intuitiv repräsentiert G die Ziele, die noch zu lösen sind, und C die bereits aufgetretenen, vereinfachten Constraints.

Entfalten:

$$\frac{(B \leftarrow H) \text{ aus } P \qquad CT \models \exists((B \doteq A) \wedge C)}{\langle A \wedge G, C\rangle \mapsto_{Entfalten} \langle H \wedge G, (B \doteq A) \wedge C\rangle}$$

Scheitern:

Es gibt keine Klausel $B \leftarrow H$ aus P mit

$$\frac{CT \models \exists((B \doteq A) \wedge C)}{\langle A \wedge G, C\rangle \mapsto_{Scheitern} \langle \bot, \mathit{false}\rangle}$$

Vereinfachen:

$$\frac{CT \models (C \wedge D_1) \leftrightarrow D_2}{\langle C \wedge G, D_1\rangle \mapsto_{Vereinfachen} \langle G, D_2\rangle}$$

Anmerkung

Im folgenden ist $p(t_1, \ldots, t_n) \doteq p(s_1, \ldots, s_n)$ eine Abkürzung für $t_1 \doteq s_1 \wedge \ldots \wedge t_n \doteq s_n$.

Die Kongruenz zwischen Konjunktionen von Zielen ist dieselbe wie im LP-Kalkül. Die Auswahl der Ziele innerhalb eines Zustandes wird wieder durch die Selektionsfunktion bestimmt. Die meisten CLP-Sprachen benutzen wie Prolog die Tiefensuche mit chronologischem Rücksetzen. Die Reduktionen des CLP-Kalküls wurden so gewählt, daß die Ähnlichkeit zu denen des LP-Kalküls deutlich wird.

CLP-Kalkül

Definition 9 Jedem Zustand $\langle H, C\rangle$, der in einer Ableitung beginnend mit $\langle G, true\rangle$ vorkommt, wird eine Formel $\exists\bar{x}(H \wedge C)$ als *logische Beschreibung* zugeordnet, wobei $\bar{x}$ die Variablen sind, die in H oder C, aber nicht in G vorkommen. Die in G vorkommenden Variablen bleiben frei.

Ein *Antwortconstraint (kurz: eine Antwort) eines Zieles* G ist die logische Beschreibung eines Endzustandes einer Ableitung, die mit $\langle G, true\rangle$ beginnt. $\lhd$

Anmerkung

Ein Unterschied zwischen LP und CLP ist die Gestalt der berechneten Antworten. Eine Antwort ist in LP-Sprachen eine Variablenbindung, in CLP-Sprachen ein Constraint. Constraints als Antworten sind sinnvoll, da sie mehrere (manchmal unendlich viele) LP-Antworten zu einer Antwort zusammenfassen (intensionale Antworten). Das Ziel $X+Y \geq 3 \wedge X+Y \leq 3$ z. B. wird zur intensionalen Antwort $X + Y \doteq 3$ vereinfacht, Variablen werden aber nicht gebunden.

CLP Beispiel Minimum 1

$\{\texttt{min(X,Y,Z)} \leftarrow \texttt{X} \leq \texttt{Y} \wedge \texttt{X} \doteq \texttt{Z},$
$\texttt{min(X,Y,Z)} \leftarrow \texttt{Y} \leq \texttt{X} \wedge \texttt{Y} \doteq \texttt{Z} \}$

$$\begin{array}{ll} & \langle \texttt{min}(1,2,\texttt{C}), \mathit{true} \rangle \\ \mapsto_{Entfalten} & < \texttt{X} \leq \texttt{Y} \wedge \texttt{X} \doteq \texttt{Z},\ 1 \doteq \texttt{X} \wedge 2 \doteq \texttt{Y} \wedge \texttt{C} \doteq \texttt{Z} > \\ \mapsto_{Vereinfachen} & < \top, \texttt{C} \doteq 1 > \end{array}$$

Anmerkung

Das folgende CL-Programm implementiert das Prädikat `min/3`. `min(X,Y,Z)` trifft genau dann zu, wenn `Z` das Minimum von `X` und `Y` ist:

Hierbei erfolgt zuerst eine Reduktion *Entfalten*, bei der die erste Klausel ausgewählt wird. Dann erfolgt eine Constraintvereinfachung durch Anwendung der Reduktion *Vereinfachen*.

Die Ableitung ist erfolgreich und das Antwortconstraint ist $\texttt{C} \doteq \texttt{1}$. In den Beispielen führen wir beim Vereinfachen auch eine Projektion durch, um ein einfaches Antwortconstraint zu erhalten.

Betrachten wir nun eine andere Ableitung ausgehend vom obigen Anfangszustand. Die Auswahl der zweiten Klausel im ersten Ableitungsschritt führt zu einem inkonsistenten Constraintspeicher, nämlich $\texttt{2} \leq \texttt{1} \wedge \texttt{2} \doteq \texttt{C}$. Damit scheitert diese Ableitung.

CLP Beispiel Minimum 2

$$\langle \texttt{min(A,2,1)}, \mathit{true} \rangle$$

$$\mapsto_{Entfalten} \quad \texttt{<X}\leq\texttt{Y}\wedge\texttt{X}\doteq\texttt{Z, A}\doteq\texttt{X}\wedge\texttt{2}\doteq\texttt{Y}\wedge\texttt{1}\doteq\texttt{Z>}$$

$$\mapsto_{Vereinfachen} \quad \texttt{<} \top\texttt{, A}\doteq\texttt{1>}$$

$$\langle \texttt{min(A,A,B)}, \mathit{true} \rangle$$

$$\mapsto_{Entfalten} \quad \texttt{<X}\leq\texttt{Y}\wedge\texttt{X}\doteq\texttt{Z, A}\doteq\texttt{X}\wedge\texttt{A}\doteq\texttt{Y}\wedge\texttt{B}\doteq\texttt{Z>}$$

$$\mapsto_{Vereinfachen} \quad \texttt{<} \top\texttt{, A}\doteq\texttt{B>}$$

Anmerkung

Das Ziel `min(A,2,2)` führt zur Antwort `A` $\doteq$ `2`, wobei mit jeder der beiden Klauseln eine Ableitung mit dieser Antwort möglich ist. Das Ziel `min(A,2,3)` scheitert.

In Prolog würden diese Reduktionen zu einer Fehlermeldung führen, da noch kein Wert für `A` bekannt ist und der Vergleich nur zwischen bekannten Werten möglich ist. Im CLP-Kalkül ist es aber möglich, auch hier „rückwärts“ zu rechnen.

Die intensionale Antwort sagt uns, daß das Minimum von zwei gleichen Zahlen diese Zahl selbst ist. Die Auswahl der zweiten Klausel führt zum selben Endzustand.

Stabilitätseigenschaft für CLP-Sprachen

Wenn $\langle G, C\rangle \mapsto_{Entfalten} \langle G', C'\rangle$ und D ist ein Constraint und $C' \wedge D$ ist erfüllbar, dann ist $\langle G \wedge H, C \wedge D\rangle \mapsto_{Entfalten} \langle G' \wedge H, C' \wedge D\rangle$ möglich.

Anmerkung

Wenn eine Reduktion *Entfalten* zu einem Folgezustand mit Constraintspeicher C' führt, dann kann diese Reduktion auch auf jeden Zustand mit zusätzlichem Ziel und speziellerem Constraintspeicher als C' (solange er konsistent bleibt) angewendet werden, und der Folgezustand unterscheidet sich nur durch das zusätzliche Ziel und das zusätzliche Constraint.

Deklarative Semantik

Theorem 1 Korrektheit und Vollständigkeit von erfolgreichen Ableitungen [JaffarLassez87] Sei P ein CL-Programm und G ein Ziel. Wenn G eine erfolgreiche Ableitung mit einem Antwortconstraint C hat, dann gilt

$$P^{\leftrightarrow}, CT \models \forall C \rightarrow G$$

Wenn $P^{\leftrightarrow}, CT \models \forall(C \rightarrow G)$ gilt und C in CT erfüllbar ist, dann hat G erfolgreiche Ableitungen mit Antwortconstraints $C_1, \ldots, C_n$, so daß

$$CT \models \forall C \rightarrow (C_1 \vee \ldots \vee C_n).$$

Anmerkung

Der Vollständigkeitssatz für CLP zeigt, daß man im Gegensatz zum LP-Kalkül mehrere Antworten betrachten und kombinieren muß, um die Vollständigkeit zu erhalten. Falls aber das Constraintsystem unabhängig ist, kann die Disjunktion zu einem Disjunkt reduziert werden.

Deklarative Semantik

Beispiel 1 Sei P das folgende Programm und CT die Constrainttheorie, die u.a. die Constraints $\leq$ und $\geq$ interpretiert:

$\texttt{p(X,Y)} \leftarrow \texttt{X} \leq \texttt{Y}.$
$\texttt{p(X,Y)} \leftarrow \texttt{X} \geq \texttt{Y}.$

Das Ziel `p(X,Y)` hat zwei erfolgreiche Ableitungen mit den Antwortconstraints $\texttt{X} \leq \texttt{Y}$ bzw. $\texttt{X} \geq \texttt{Y}$.

Es gilt $P^{\leftrightarrow}, CT \models \forall(true \rightarrow \texttt{p}(\texttt{X}, \texttt{Y}))$; nach dem Satz also auch $CT \models \forall(true \rightarrow \texttt{X} \leq \texttt{Y} \vee \texttt{X} \geq \texttt{Y})$.

Jedes Antwortconstraint für sich allein reicht aber nicht aus: $CT \not\models \forall(true \rightarrow \texttt{X} \leq \texttt{Y})$ und $CT \not\models \forall(true \rightarrow \texttt{X} \geq \texttt{Y})$. $\diamond$

Anmerkung

Deklarative Semantik

Theorem 2 Vollständigkeit und Korrektheit von erfolglosen Ableitungen
Sei P ein CL-Programm und G ein Ziel. Es gilt $P^{\leftrightarrow}, CT \models \neg\exists G$ genau dann, wenn jede faire Ableitung beginnend mit $\langle G, true\rangle$ endlich scheitert.

Anmerkung