Thom Frühwirth, Slim Abdennadher, LMU München

Constrainterweiterungen

Anmerkung

Diese Erweiterungen basieren auf der Idee, daß man nicht nur Konjunktionen von Constraintatomen, sondern beliebige Formeln von Constraints erlauben und behandeln können möchte.

Für die vorgestellten Constrainterweiterungen[1] reicht es aus, wenn der Constraintlöser den Folgerungstest implementiert.

Minimum-Berechnung bietet sich an, da es ausschließlich durch Constraints definierbar ist, als $min(X, Y, Z) \leftrightarrow (X \leq Y \wedge X \doteq Z) \vee (Y \leq X \wedge Y \doteq Z)$. Im Gegensatz zur Implementierung mit CL-Klauseln erspart uns die Formulierung von Minimum mit Constrainterweiterungen die Suche (nach der passenden Klausel) auf Kosten der Vollständigkeit. Der Suchaufwand für Minimum mag vernachläßigbar erscheinen gegenüber den Einbußen bei der Vollständigkeit. Das folgende Anwendungsbeispiel widerlegt aber diese Vermutung: Wir suchen das kleinste Element einer Liste.

[1] mit Ausnahme der konstruktiven Disjunktion

Implikation

Die Implikation hat die Form $C \Longrightarrow D$ und wird als "wenn C dann D" gelesen. Die deklarative Semantik der Implikation $C \Longrightarrow D$ ist die logische Implikation $C \rightarrow D$.

Erweiterte Syntax von Zielen

Ziele: $G, H \ ::= \ \top \mid \bot \mid A \mid C \mid G \wedge H \mid \ C \Longrightarrow D$

Reduktionsregeln von Implikationen

Positive Erfüllung

$$\frac{CT \models \forall(C_0 \rightarrow C)}{\langle (C \Longrightarrow D) \wedge G, C_0 \rangle \mapsto \langle D \wedge G, C_0 \rangle}$$

Negative Erfüllung

$$\frac{CT \models \forall(C_0 \rightarrow \neg C)}{\langle (C \Longrightarrow D) \wedge G, C_0 \rangle \mapsto \langle G, C_0 \rangle}$$

Anmerkung

Wir erweitern das CLP-Kalkül um die von Saraswat vorgeschlagene Implikation zwischen Constraints.

Wir nennen C die *(Vor-)Bedingung* und D das *bedingte Constraint* der Implikation $C \Longrightarrow D$.

Operational stellt die Implikation $C \Longrightarrow D$ sicher, daß das Constraint D erst und nur dann eingefügt wird, wenn das Constraint C folgt. Wenn dagegen $\neg C$ folgt, wird D nicht eingefügt. Wenn weder C noch $\neg C$ folgt, wird das Konstrukt verzögert bis der Folgerungstest entschieden werden kann.

Die Reduktionsregeln sind (bewußt) unvollständig, z. B. wird $C \Longrightarrow \mathit{false}$ nicht zum logisch äquivalenten $\neg C$ vereinfacht[1].

[1]Man kann übrigens mit $C \Longrightarrow \mathit{false}$ eine korrekte (wenn auch unvollständige) Form der Negation von Constraints implementieren.

Implikation

- Das Ziel $\mathtt{X} \doteq -4 \land (\mathtt{X} \leq 0 \Longrightarrow \mathtt{X} \doteq \mathtt{Y})$ führt zur Antwort $\mathtt{X} \doteq -4 \land \mathtt{Y} \doteq -4$.

- Das Ziel $\mathtt{X} \doteq 4 \land (\mathtt{X} \leq 0 \Longrightarrow \mathtt{X} \doteq \mathtt{Y})$ führt zu $\mathtt{X} \doteq 4$.

- Das Ziel $\mathtt{X} \leq -4 \land (\mathtt{X} \leq 0 \Longrightarrow \mathtt{X} \doteq \mathtt{Y})$ führt zu $\mathtt{X} \leq -4 \land \mathtt{X} \doteq \mathtt{Y}$, denn aus $\mathtt{X} \leq -4$ folgt $\mathtt{X} \leq 0$.

- Das Ziel $\mathtt{X} \leq 4 \land (\mathtt{X} \leq 0 \Longrightarrow \mathtt{X} \doteq \mathtt{Y})$ kann nicht vereinfacht werden, weil aus $\mathtt{X} \leq 4$ weder $\mathtt{X} \leq 0$ noch die Negation von $\mathtt{X} \leq 0$ folgt.

Anmerkung

Wird eine Implikation *verzögert*, dann wird die Ableitung mit einem anderen Zielatom fortgesetzt, solange, bis die Vorbedingung der Implikation entschieden werden kann. Dies bedeutet, daß wir eine einfache Form von *Nebenläufigkeit* in den um die Implikation erweiterten CLP-Sprachen erhalten: Im Laufe der Ableitung gibt es i. a. mehrere Implikationen, die verzögert sind und auf mehr Information im Constraintspeicher warten. Damit durchbrechen wir die feste Selektionsregel der SLDNF-Resolution.

Implikation

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

Das Ziel `min(1,2,C)` entfaltet zu $(\texttt{1} \leq \texttt{2} \Longrightarrow \texttt{1}\doteq\texttt{C}) \wedge (\texttt{2} \leq \texttt{1} \Longrightarrow \texttt{2}\doteq\texttt{C})$. Die Antwort ist $\texttt{C}\doteq\texttt{1}$.

Das Ziel `min(A,2,1)` führt zur Antwort $(\texttt{A} \leq \texttt{2} \Longrightarrow \texttt{A}\doteq\texttt{1}) \wedge (\texttt{2} \leq \texttt{A} \Longrightarrow \texttt{2}\doteq\texttt{1})$. Wir können im Gegensatz zur CLP-Formulierung des Prädikates `min/3` nicht mehr durch Suche die Antwort $\texttt{A} \doteq \texttt{1}$ finden.

Das Ziel `min(A,A,B)` führt zu $\texttt{A} \doteq \texttt{B}$. Die Bedingungen beider Implikationen folgen.

Das Ziel $\texttt{min}(\texttt{A},\texttt{B},\texttt{C}) \wedge \texttt{A} \leq \texttt{B}$ führt zu $\texttt{A} \leq \texttt{B} \wedge \texttt{A}\doteq\texttt{C}$, während es in der CLP-Formulierung noch eine zweite, redundante Antwort mit $\texttt{A} \doteq \texttt{B}$ gab.

Anmerkung

Die Implikationen drücken aus, daß wir das Minimum von `X` und `Y` erst berechnen wollen, wenn wir wissen, wie sich die beiden Variablen zueinander verhalten. Damit erreichen wir, daß `min/2` nun deterministisch ist, also ohne Suche auskommt. Wir verlieren aber dafür an Vollständigkeit. Allerdings ist dies nur der Fall, wenn sich im Lauf der Berechnung nicht genügend Information über die beiden ersten Argumente im Constraintspeicher ansammelt, wie die folgenden Beispiele zeigen. Es kommt also auf die Verwendung von `min` im Programm und Effizienzüberlegungen an, welcher Version man den Vorzug gibt.

Konditional

Reduktionsregeln von Konditionalen

Positive Erfüllung

$$\frac{CT \models \forall (D \rightarrow C)}{\langle (C \Longrightarrow G_1; G_2) \wedge H, D\rangle \mapsto \langle G_1 \wedge H, D\rangle}$$

Negative Erfüllung

$$\frac{CT \models \forall (D \rightarrow \neg C)}{\langle (C \Longrightarrow G_1; G_2) \wedge H, D\rangle \mapsto \langle G_2 \wedge H, D\rangle}$$

Anmerkung

Eine Erweiterung der Implikation ist das Konditional von Smolka. Es kann als deklarative Version der bedingten Anweisung (auch: Verzweigung, engl. “if-then-else”) herkömmlicher Programmiersprachen verstanden werden. Das Konditional hat die Form $C \implies G_1; G_2$ und wird als “wenn C dann G_1 sonst G_2” gelesen. Wir nennen G_1 und G_2 *Zweige* des Konditionals. Die deklarative Semantik des Konditionals $C \implies G_1; G_2$ ist $(C \wedge G_1) \vee (\neg C \wedge G_2)$. Das Konditional ist aber keine Constrainterweiterung im eigentlichen Sinne, da allgemeine Ziele G_1 und G_2 als Zweige zugelassen sind.

Die Reduktionsregeln sind analog zu denen der Implikation. Wenn C folgt, dann ersetzt man das Konditional mit *Positive Erfüllung* durch G_1. Wenn $\neg C$ folgt, dann ersetzt man das Konditional mit *Negative Erfüllung* durch G_2. Andernfalls wird das Konditional verzögert.

Konditional

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

Im Vergleich zur Implementierung mit der Implikation gibt es nur mehr eine Vorbedingung, die überprüft werden muß. Es mag scheinen, daß sich die beiden Implementierungen operational gleich verhalten. Die Implementierung von `min` mit dem Konditional ist aber weniger vollständig als die Version mit Implikation: Das Ziel $\texttt{min}(\texttt{A}, \texttt{B}, \texttt{C}) \wedge \texttt{A} \geq \texttt{B}$ führt bloß zur Antwort $(\texttt{A} \leq \texttt{B} \Longrightarrow \texttt{A} \doteq \texttt{C}; \texttt{B} \doteq \texttt{C}) \wedge \texttt{A} \geq \texttt{B}$, während das Ziel $\texttt{min}(\texttt{A}, \texttt{B}, \texttt{C}) \wedge \texttt{A} \leq \texttt{B}$ erwartungsgemäß zu $\texttt{A} \leq \texttt{B} \wedge \texttt{A} \doteq \texttt{C}$ führt.

Keine Anmerkung

Disjunktion

Positive Durchführung

$$\langle G \wedge \bigvee_{i=1}^{1} C_i, C\rangle \mapsto \langle G \wedge C_1, C\rangle$$

Positive Erfüllung

$$\frac{CT \models \forall(C \rightarrow C_j) \quad \text{für ein } j \in \{1,\ldots,n\}}{\langle G \wedge \bigvee_{i=1}^{n} C_i, C\rangle \mapsto \langle G, C\rangle}$$

Negative Reduktion

$$\frac{CT \models \forall(C \rightarrow \neg C_j) \quad \text{für ein } j \in \{1,\ldots,n\}, \quad n > 1}{\langle G \wedge \bigvee_{i=1}^{n} C_i, C\rangle \mapsto \langle G \wedge (C_1 \vee \ldots \vee C_{j-1} \vee C_{j+1} \vee \ldots \vee C_n), C\rangle}$$

Anmerkung

Die Syntax von Zielen wird diesmal um die Disjunktion von Constraints erweitert, geschrieben $C_1 \vee \ldots \vee C_n$ (mit $n > 0$), abgekürzt $\bigvee_{i=1}^{n} C_i$ layout. Man betrachtet Disjunkte, von denen man entscheiden kann, ob sie oder ihre Negation folgen. Ein Disjunkt, das folgt, läßt die ganze Disjunktion folgen (durch die Reduktionsregel *Positive Erfüllung*). Ein Disjunkt, dessen Negation folgt, kann aus der Disjunktion entfernt werden (*Negative Reduktion*). Hat man schließlich nur mehr ein Disjunkt übrig, so muß es erfüllt werden, damit es folgt (*Positive Durchführung*).

Eine weniger vollständige Variante der Disjunktion erhält man übrigens, wenn man *Positive Durchführung* durch eine Reduktionsregel ersetzt, die eine Disjunktion ohne Disjunkte ($n = 0$) scheitern läßt, und die Vorbedingung $n > 1$ der Reduktionsregel *Negative Reduktion* wegläßt.

Disjunktion

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

Das Ziel `min(A,A,B)` führt aber lediglich zur Antwort $(\texttt{A} \leq \texttt{A} \wedge \texttt{A} \doteq \texttt{B}) \vee (\texttt{A} \leq \texttt{A} \wedge \texttt{A} \doteq \texttt{B})$ und kann nicht weiter vereinfacht werden, da keines der beiden (wiewohl identischen) Disjunkte für sich allein folgt.

Analog für die Ziele `min(A,2,2)`, $\texttt{min}(\texttt{A}, \texttt{B}, \texttt{C}) \wedge \texttt{A} \leq \texttt{B}$ und $\texttt{min}(\texttt{A}, \texttt{B}, \texttt{C}) \wedge \texttt{A} \geq \texttt{B}$.

Anmerkung

Die Ziele `min(1,2,C)`, `min(A,2,1)` und `min(A,2,3)` führen alle durch *Negative Reduktion* und anschließende *Positive Durchführung* zu den bekannten Antworten.

Konstruktive Disjunktion

Zusätzliche Reduktionsregel von konstruktiven Disjunktionen

Generalisierung

$$\frac{CT \models \forall(((C \wedge \bigvee_{i=1}^{n} C_i) \rightarrow D) \;\wedge\; \neg(C \rightarrow D))}{\langle G \wedge \bigvee_{i=1}^{n} C_i, C\rangle \mapsto \langle G \wedge \bigvee_{i=1}^{n} C_i, C \wedge D\rangle}$$

Anmerkung

Diese Variante der Disjunktion ist vollständiger, weil eine neue, mächtige Reduktionsregel zum Kalkül hinzugefügt wird. Die Idee ist, daß man aus der Disjunktion im aktuellen Kontext des Constraintspeichers, C, ein neues Constraint D gewinnen kann, das für jedes Disjunkt C_i im Kontext erfüllt sein muß. Es wird sozusagen der "kleinste gemeinsame Nenner" aller Disjunkte in Form eines Constraints ermittelt. Wir sagen auch, daß bei Anwendung der Reduktionsregel *Generalisierung* die Disjunktion das Constraint D *propagiert*. Das zweite Konjunkt $\neg(C \rightarrow D)$ in der Bedingung der Reduktionsregel stellt sicher, daß D "spezieller" als C ist. Wie stark und von welcher Art das propagierte Constraint sein kann, hängt von Effizienzüberlegungen ab. I. a. ist die *Generalisierung* sehr aufwendig zu implementieren.

Die Reduktionsregel *Positive Durchführung* kann nun entfallen, da sie sich als Spezialfall der *Generalisierung* mit nachfolgender *Positive Erfüllung* herausstellt.

Konstruktive Disjunktion

Beispiel 1 Das folgende Constraint könnte ohne die Reduktionsregel *Generalisierung* nicht vereinfacht werden: $(\mathtt{X} < \mathtt{Y} \vee \mathtt{X} < \mathtt{Z}) \wedge \mathtt{Y} < \mathtt{Z}$ propagiert mittels *Generalisierung* das Constraint $\mathtt{X} < \mathtt{Z}$.
$(\mathtt{X} < \mathtt{Y} \vee \mathtt{X} < \mathtt{Z}) \wedge \mathtt{Y} < \mathtt{Z} \wedge \mathtt{X} < \mathtt{Z}$ führt nun durch *Positive Erfüllung* mit dem Disjunkt $\mathtt{X} < \mathtt{Z}$ zur Antwort $\mathtt{Y} < \mathtt{Z} \wedge \mathtt{X} < \mathtt{Z}$. $\diamond$

Keine Anmerkung

Konstruktive Disjunktion

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

Die Disjunktion $(\mathtt{A} \leq \mathtt{A} \wedge \mathtt{A}\doteq\mathtt{B}) \vee (\mathtt{A} \leq \mathtt{A} \wedge \mathtt{A}\doteq\mathtt{B})$ kann durch *Generalisierung* vereinfacht werden. Sie propagiert trivialerweise $(\mathtt{A}\doteq\mathtt{B})$. Danach folgen beide Disjunkte und *Positive Erfüllung* ist anwendbar.

Das Ziel `min(A,2,2)` führt zur Disjunktion $(\mathtt{A} \leq \mathtt{2} \wedge \mathtt{A}\doteq\mathtt{2}) \vee (\mathtt{2} \leq \mathtt{A} \wedge \mathtt{2}\doteq\mathtt{2})$ und diese propagiert das Constraint $2 \leq A$. Daraus folgt das zweite Disjunkt.

Das Ziel $\mathtt{min}(\mathtt{A}, \mathtt{B}, \mathtt{C}) \wedge \mathtt{A} \leq \mathtt{B}$ führt zur Propagierung von $\mathtt{A}\doteq\mathtt{C}$. Zusammen mit $\mathtt{A} \leq \mathtt{B}$ folgt daraus trivialerweise das erste Disjunkt $\mathtt{A} \leq \mathtt{B} \wedge \mathtt{A}\doteq\mathtt{C}$.

Anmerkung

Bereits aus dem Rumpf der Klausel kann durch die Anwendung der *Generalisierung* gefolgert werden, daß in jedem Fall $\texttt{Z} \leq \texttt{X} \wedge \texttt{Z} \leq \texttt{Y}$ erfüllt sein muß, d. h., daß das Minimum zweier Zahlen auf jeden Fall kleiner oder gleich als jede der beiden Zahlen ist.

Kardinalität

Folgende Spezialfälle der Kardinalität entsprechen den erwähnten logischen Verknüpfungen:

- $\#(0, [C], 0)$ entspricht der Negation des Constraints, $\neg C$.
- $\#(n, [C_1, \ldots, C_n], n)$ entspricht der Konjunktion der Constraints, $\bigwedge\limits_{i-1}^{n} C_i$.
- $\#(1, [C_1, \ldots, C_n], n)$ entspricht der Disjunktion der Constraints, $\bigvee\limits_{i=1}^{n} C_i$.

Anmerkung

Die Kardinalität (vorgeschlagen von Van Hentenrynck) verallgemeinert die logischen Verknüpfungen Negation, Disjunktion und Konjunktion für Constraints, indem er die Anzahl der Constraints, die folgen, zählt. Der Ausdruck $\#(L, [C_1, \ldots, C_n], U)$ bedeutet, daß zwischen L und maximal U Constraints in der Aufzählung $[C_1, \ldots, C_n]$ wahr sind.

Reductionsregeln von Kardinalitäten

Triviale Erfüllung

$$\frac{L \leq 0 \land U \geq n}{\langle \#(L, K, U) \land G, C\rangle \mapsto \langle G, C\rangle}$$

Positive Durchführung

$$\frac{L = n}{\langle \#(L, K, U) \land G, C\rangle \mapsto \langle C_1 \land \ldots \land C_n \land G, C\rangle}$$

Negative Durchführung

$$\frac{U = 0}{\langle \#(L, K, U) \land G, C\rangle \mapsto \langle \neg C_1 \land \ldots \land \neg C_n \land G, C\rangle}$$

Positive Reduktion

$$\frac{CT \models \forall(C \rightarrow C_j) \quad \text{für ein } j \in \{1, \ldots, n\} \quad U > 0}{\langle \#(L, K, U) \land G, C\rangle \mapsto \langle \#(L-1, [\ldots, C_{j-1}, C_{j+1}, \ldots], U-1) \land G, C\rangle}$$

Negative Reduktion

$$\frac{CT \models \forall(C \rightarrow \neg C_j) \quad \text{für ein } j \in \{1, \ldots, n\} \quad L < n}{\langle \#(L, K, U) \land G, C\rangle \mapsto \langle \#(L, [\ldots, C_{j-1}, C_{j+1}, \ldots], U) \land G, C\rangle}$$

Anmerkung

Die operationale Semantik der Kardinalität ist durch folgende Reduktionen definiert, wobei wir voraussetzen: $L \leq U, 0 \leq U, L \leq n$. K steht für $[C_1, \ldots, C_n]$.

Die ersten drei Reduktionsregeln behandeln Spezialfälle: Sind die Grenzen L und U zu weit, ist die Kardinalität erfüllt (*Triviale Erfüllung*). Ist $L = n$, müssen alle Constraints erfüllt werden (*Positive Durchführung*). Ist $U = 0$, darf keines der Constraints erfüllbar sein (*Negative Durchführung*). *Positive Reduktion* und *Negative Reduktion* entfernen ein Constraint, das folgt bzw. dessen Negation folgt, und dekrementieren im positiven Fall die Grenzen L und U.

Die Reduktionsregel *Negative Durchführung* kann nur implementiert werden, wenn negierte Constraints erlaubt sind. Andernfalls ist eine weniger vollständige Variante durch eine Reduktionsregel möglich, die, falls $U = 0$ ist und ein C_i folgt, zu einem gescheiterten Endzustand führt.

Kardinalität

Beispiel 2 Die Kardinalität $\#(\texttt{1}, [\texttt{X}{=}\texttt{4}, \texttt{Y}{=}\texttt{10}], \texttt{2})$ besagt, daß $X \doteq 4$ oder $Y \doteq 10$ erfüllt werden muß. Kommt das Constraint $X > 6$ hinzu, folgt daraus $\neg(X \doteq 4)$. Durch *Negative Reduktion* wird die Kardinalität zu $\#(\texttt{1}, [\texttt{Y}{=}\texttt{10}], \texttt{2})$ vereinfacht. Durch *Positive Durchführung* wird die Kardinalität zu $\texttt{Y}{=}\texttt{10}$ vereinfacht.

Das Prädikat `min` kann durch eine Kardinalität $\#(\texttt{1}, [(\texttt{X} \leq \texttt{Y} \wedge \texttt{X}{=}\texttt{Z}),\ (\texttt{Y} \leq \texttt{X} \wedge \texttt{Y}{=}\texttt{Z})], \texttt{2})$ implementiert werden, das Verhalten ist daher dasselbe wie bei der Disjunktion ohne Generalisierung. $\diamond$

Keine Anmerkung

Reductionsregeln von Meta-Constraints

Positive Erfüllung

$$\frac{CT \models \forall(D \rightarrow C)}{\langle (C \Longleftrightarrow X) \wedge H, D\rangle \mapsto \langle X \doteq 1 \wedge H, D\rangle}$$

Negative Erfüllung

$$\frac{CT \models \forall(D \rightarrow \neg C)}{\langle (C \Longleftrightarrow X) \wedge H, D\rangle \mapsto \langle X \doteq 0 \wedge H, D\rangle}$$

Positive Durchführung

$$\frac{CT \models \forall(D \rightarrow X \doteq 1)}{\langle (C \Longleftrightarrow X) \wedge H, D\rangle \mapsto \langle C \wedge H, D\rangle}$$

Negative Durchführung

$$\frac{CT \models \forall(D \rightarrow X \doteq 0)}{\langle (C \Longleftrightarrow X) \wedge H, D\rangle \mapsto \langle \neg C \wedge H, D\rangle}$$

Anmerkung

In der englischsprachigen Literatur gibt es noch keine einheitliche Nomenklatur für dieses Konstrukt, das wir Meta-Constraint nennen und das in den letzten Jahren in verschiedenen Varianten in immer mehr Implementierungen aufgetaucht ist, es wird dort “nested constraint”, “meta constraint” oder auch “reified constraint” genannt.

Das Meta-Constraint hat die Form $C \Longleftrightarrow X$ und verbindet das Constraint C mit einer Variable. Die Variable X hat den Wert 1, wenn C folgt, und sie hat den Wert 0, wenn $\neg C$ folgt. Umgekehrt muß C erfüllt werden, wenn $X \doteq 1$ folgt, und $\neg C$ erfüllt werden, wenn $X \doteq 0$ folgt. Die logische Leseweise ist also $(C \leftrightarrow (X \doteq 1)) \wedge (\neg C \leftrightarrow (X \doteq 0))$.

Die Reduktionsregel *Negative Durchführung* kann wieder nur implementiert werden, falls der Constraintlöser negierte Constraints erlaubt. Läßt man sie weg, erhält man eine weniger vollständige Variante des Meta-Constraints.

Meta-Constraint

Beispiel 3 Das Constraint
$(\mathtt{A}\dot{=}-\mathtt{1} \vee \mathtt{A}\dot{=}\mathtt{1}) \wedge \mathtt{A} < \mathtt{0}$ läßt sich durch das Ziel
$((\mathtt{X} \sqcup \mathtt{Y}{=}_{\mathtt{B}}\mathtt{1}) \ \wedge \ (\mathtt{A}\dot{=}-\mathtt{1} \Longleftrightarrow \mathtt{X}) \ \wedge \ (\mathtt{A}\dot{=}\mathtt{1} \Longleftrightarrow \mathtt{Y})) \ \wedge \ \mathtt{A} < \mathtt{0})$
ausdrücken. Da $A < 0$ ist, folgt $\neg(\mathtt{A}\dot{=}\mathtt{1})$. Das Meta-Constraint $(\mathtt{A}\dot{=}\mathtt{1} \Longleftrightarrow \mathtt{Y})$ wird daher mit *Negative Erfüllung* ersetzt durch $\mathtt{Y}{=}_{\mathtt{B}}\mathtt{0}$. Damit kann der Boolesche Constraintlöser das Constraint $\mathtt{X} \sqcup \mathtt{Y}{=}_{\mathtt{B}}\mathtt{1}$ zu $\mathtt{X}\dot{=}_{\mathtt{B}}\mathtt{1}$ vereinfachen. Dies bewirkt, daß das Meta-Constraint $\mathtt{A}\dot{=}-\mathtt{1} \Longleftrightarrow \mathtt{X}$ mit *Positive Durchführung* zu $\mathtt{A}{=}-\mathtt{1}$ vereinfacht wird. So erhalten wir die Antwort $\mathtt{A}\dot{=}-\mathtt{1} \ \wedge \ \mathtt{X}{=}_{\mathtt{B}}\mathtt{1} \ \wedge \ \mathtt{Y}{=}_{\mathtt{B}}\mathtt{0}$. $\diamond$

Anmerkung

Mit dem Meta-Constraint läßt sich *jede* logische Formel über Constraints (zumindest unvollständig) implementieren, vorausgesetzt, der Constraintlöser stellt auch einen Löser für Boolesche Constraints zu Verfügung (siehe Kapitel **??**). Boolesche Constraints sind logische Formeln über Variablen, die nur die Werte 1 oder 0 annehmen können. $=_{\mathtt{B}}$ steht für Gleichheit von Booleschen Termen, $\sqcup$ für Disjunktion.

Wir können übrigens die Disjunktion, $\bigvee_{i=1}^{n} C_i$, mit Meta-Constraints implementieren. Wir schreiben ein Meta-Constraint $C_i \iff X_i$ für jedes Constraint C_i aus der Disjunktion. Die Disjunktion $\bigvee_{i=1}^{n} C_i$ ist jetzt genau dann erfüllt, wenn zusätzlich das Boolesche Gleichheitsconstraint $(X_1 \sqcup \ldots \sqcup X_n) =_B 1$ erfüllt ist.