Kapitel 5

Set Constraints

5.1 Motivation

Ein deklaratives (logisches oder funktionales) Programm ist ein Gleichungssystem G. Wir suchen dessen Lösung, d. h. eine Belegung der darin vorkommenden Variablen, wenn eine partielle Belegung (der Eingabevariablen) gegeben ist. Belegungen bilden Variablen auf Grundterme ab.

Wir ersetzen in G jede Variable x durch eine Mengenvariable X mit der beabsichtigten Interpretation "alle Werte, die x annehmen kann".

Wir erhalten ein Gleichungssystem G' für Mengenausdrücke, Eigenschaften solcher Systeme sind – unter Benutzung von Baum-Automaten – entscheidbar, selbst das für G nicht zutrifft.

Das liegt daran, daß wir beim Übergang von G zu G' Information verlieren: in $f(X) = Y$ wissen wir nicht, welches $x \in X$ dafür sorgt, daß $y \in Y$ ist.

Wir können Typ-Inferenz für Programme als Lösen solcher Systeme auffassen.

5.2 Definition

Def. Sei Ω die Signatur mit $\Omega_0 = \{\top, \bot\}, \Omega_1 = \{\neg\}, \Omega_2 = \{\cup, \cap\}$. Für eine Signatur Σ und eine Menge von Variablen V ist ein Set-Constraint-System über (Σ, V) eine endliche Menge von Ausrücken der Form $l \subseteq r$ und $l \not\subseteq r$ mit $l, r \in \mathsf{Tree}(\Sigma \cup \Omega, V)$. Die Menge der Set-Constraint-Systeme über (Σ, V) heißt $\mathsf{SCS}(\Sigma, V)$.

Bsp. $\{Y \not\subseteq \bot, X \subseteq f(Y, \neg X) \cup a\}$.

Def. Eine Belegung

$$B : V \to 2^{\mathsf{Tree}(\Sigma)}$$

von Variablen mit Mengen von Bäumen läßt sich ausdehnen zu einer Bewertung von Termen

$$B : \mathsf{Tree}(\Sigma \cup \Omega, V) \to 2^{\mathsf{Tree}(\Sigma)},$$

durch

$$\begin{aligned} B(\top) = \mathsf{Tree}(\Sigma), B(\bot) = \emptyset \\ B(\neg t) = \mathsf{Tree}(\Sigma) \setminus B(t) \\ B(t \cup s) = B(t) \cup B(s), B(t \cap s) = B(t) \cap B(s) \\ B(f(t_1, \ldots, t_k)) = f(B(t_1), \ldots, B(t_k)) \end{aligned}$$

und schließlich zu einer Bewertung von Constraints

$$B : \mathsf{SCS}(\Sigma, V) \to \mathsf{Bool}$$

vermittels

$$\begin{aligned} B(\{c_1, \ldots, c_n\}) = B(c_1) \wedge \ldots \wedge B(c_n) \\ B(l \subseteq r) = B(l) \subseteq B(r), B(l \not\subseteq r) = B(l) \not\subseteq B(r). \end{aligned}$$

Def. Eine Lösung von $s \in \mathsf{SCS}(\Sigma, V)$ ist eine Belegung $B : V \to 2^{\mathsf{Tree}(\Sigma)}$ mit $B(s) = \mathsf{True}$. Die Menge aller Lösungen von s heißt $\mathsf{Sol}(s)$.

Wir zeigen, daß Lösbarkeit von SCS entscheidbar ist, und benutzen dazu verallgemeinerte Baum-Mengen-Automaten.

5.3 Beispiele für Set Constraints

keine Lösung: $X \subseteq fY, X \subseteq gZ$.

genau eine Lösung: $ffX \cup a = X$ (wir übersetzen $A = B$ in die Constraints $A \subseteq B \wedge B \subseteq A$). Die Lösung ist $X = \{f^{2n}a \mid n \geq 0\}$. Wir schreiben hierfür $X = f^{2n}a$.

mehrere Lösungen: $ffX \cup a \subseteq X$. Lösungen sind $X_0 = f^{2n}a$ aber auch $X_{infty} = f^n a$. Im Allgemeinen $X_n = \{f^{2k} \mid k \leq n\} \cup \{f^k \mid k \geq 2n\}$. Alle Lösungen sind regulär.

Wir betrachten die Halbordung "komponentenweise Inklusion" auf Lösungen:

Def. Für Belegungen $B, B' : V \to 2^{\mathsf{Tree}(\Sigma)}$ schreiben wir $B \subseteq B'$, falls $\forall v \in V : B(v) \subseteq B'(v)$.

Dann gilt in diesem Beispiel: X_0 und X_∞ sind die eindeutig bestimmten minimalen bzw. maximalen Lösungen bezüglich $\subseteq$.

hat nichtreguläre Lösungen: $fX \cup a \subseteq X, Y \subseteq X$. Eine reguläre Lösung ist $X = f^*a, Y = \emptyset$. Eine nichtreguläre Lösung ist $X = f^*a, Y = \{f^k a \mid k \text{ ist Primzahl}\}$.

mehrere maximale Lösungen: $fXY \subseteq fab$. Es gibt drei Lösungstypen:

1. $X \subseteq \{a\}, Y \subseteq \{b\}$
2. $X \subseteq \mathsf{Tree}(\Sigma), Y = \emptyset$
3. $X = \emptyset, Y \subseteq \mathsf{Tree}(\Sigma)$

Maximale Lösungen sind $(\{a\}, \{b\}), (\mathsf{Tree}(\Sigma), \emptyset), (\emptyset, \mathsf{Tree}(\Sigma))$. Einzige minimale Lösung ist $(\emptyset, \emptyset)$.

keine minimale Lösung: in der Signatur $\{f, a\}$, betrachte das System $fX \subseteq X, X \not\subseteq \emptyset$. Lösungen sind alle $X_n = f^{\geq n}a$. Wir haben $X_0 \supset X_1 \supset \ldots$, aber kein X_n ist minimal.

mehrere maximale und minimale Lösungen: $fXY \subseteq fab, X \cup Y \not\subseteq \emptyset$ Drei maximale Lösungen, wie eben, aber minimale Lösungen $(\emptyset, E), (E, \emptyset)$ für alle Einermengen E.

Das legt die Behauptung nahe: jede maximale und minimale Lösung eines Set-Constraint-Systems ist regulär (= ein Tupel regulärer Baumsprachen).

5.4 Lösen von Constraint-Systemen

Es besteht eine enge Beziehung zwischen Set Constraints und Baum-Automaten. Einerseits gilt:

Wir können jedem Baum-Automaten $A = (\Sigma, Q, Q_F, \Delta)$ ein Set-Constraint-System SCS zuordnen: jedem Zustand $q_k \in Q$ entspricht eine Mengenvariable Q_k mit der beabsichtigten Interpretation $B : Q_k \mapsto \mathsf{Lang}(A, q_k)$, und jeder Regel

$$f(q_1, \dots, q_k) \to q_0 \in \Delta$$

enstpricht ein Constraint

$$f(Q_1, \dots, Q_k) \subseteq Q_0.$$

Wir erhalten B als minimale Lösung dieses Systems, und $\mathsf{Lang}(A)$ als $B(Q)$. Wie wir sehen werden, garantiert die einfache Form der Constraints die Existenz, Eindeutigkeit und Regularität von B.

Andererseits versuchen wir, beliebige SCS mit Hilfe von Baum-Automaten zu lösen. Wir benutzen dazu:

Def. Eine Belegung (d. h. ein Tupel von Baumsprachen) heißt regulär, wenn jede Komponente eine reguläre Sprache ist. Die Menge der regulären Belegungen heißt ebenfalls REG.

Satz. Für jedes $s \in \mathsf{SCS}(\Sigma)$ gilt:
wenn $\emptyset \neq \mathsf{Sol}(s)$, dann $\exists L \in \mathsf{Sol}(s) : L \in \mathsf{REG}$.

Satz. Jede minimale und jede maximale Lösung von s ist regulär.

5.5 Klassen von Set Constraints

Wir schränken die Ausdruckskraft von Set Constraints ein, um schnellere Lösungsverfahren zu erhalten.

Andererseits versuchen wir, diese Klassen dann wieder zu erweitern, aber die Lösungsverfahren beizubehalten.

5.5.1 Positive Constraints

Def. Ein Constraint $A \subseteq B$ heißt positiv,

Def. Die Klasse PSCS enthält Set-Constraint-Systeme mit ausschließlich positiven Constraints.

Es gibt $s \in \mathsf{PSCS}$ ohne minimale Lösungen.

Satz. Erfüllbarkeit für PSCS ist entscheidbar. (Ist NEXPTIME-vollständig.)

5.5.2 Definite Constraints

Def. Ein Set Constraint $A \subseteq B$ heißt definit, wenn $A \in \mathsf{Tree}(\Sigma \cup \{\cup, \cap, \top, \bot\}, V)$ und $B \in \mathsf{Tree}(\Sigma, V)$.

D. h. es gibt keine Komplemente, und auf der rechten Seite überhaupt keine Mengenoperationen.

Def. Die Klasse DSCS enthält Set-Constraint-Systeme mit ausschließlich definiten Constraints.

Def. Für Belegungen $B, B' : V \to 2^{\mathsf{Tree}(\Sigma)}$ definieren wir komponentenweise Operationen

$$B \cup B' : V \to 2^{\mathsf{Tree}(\Sigma)} : v \mapsto B(v) \cup B'(v)$$
$$B \cap B' : V \to 2^{\mathsf{Tree}(\Sigma)} : v \mapsto B(v) \cap B'(v)$$

Satz. Für jedes $s \in \mathsf{DSCS}$ gilt: $\forall B, B' \in \mathsf{Sol}(s) : B \cup B' \in \mathsf{Sol}(s)$.

Die Menge $\mathsf{Sol}(s)$ hat deswegen ein kleinstes Element (falls sie nicht leer ist).

Satz. Erfüllbarkeit für DSCS ist entscheidbar. (Ist DEXPTIME-vollständig.)

5.5.3 Projektionen

Def. Die Signature Σ^{-1} der Projektionen einer Signatur Σ besteht aus den einstelligen Funktionssymbolen

$$\{f_i^{-1} \mid f \in \Sigma_k, 1 \leq i \leq k\}$$

Wir erweitern die Definition einer Bewertung auf

$$B : \mathsf{Tree}(\Sigma \cup \Sigma^{-1} \cup \Omega, V) \to 2^{\mathsf{Tree}(\Sigma)}$$

durch

$$B : f_i^{-1}(t) \mapsto \{t_i \mid \exists t_1, \ldots, t_k : f(t_1, \ldots, t_k) \in B(t)\}$$

Erfüllbarkeit von Set Constraints mit Projektionen ist entscheidbar. Dabei sind Constraints $f_i^{-1}(X) \subseteq Y$ einfach, aber $X \subseteq f_i^{-1}(Y)$ schwer.

Bei definiten Set Constraints können wir Projektionen auf den linken Seiten erlauben, ohne die Komplexität des Lösungsverfahrens zu erhöhen.

5.5.4 Membership-Ausdrücke

Wir erweitern linke Seiten von definiten Constraints nochmals und gestatten Mengenausdrücke mit Element-Tests, Booleschen Verknüpfungen (ohne Negation) und Quantifizierung über Bäume (d. h. es gibt eine Menge V' von Variablen erster Ordnung, zusätzlich zu den Mengenvariablen V).

Def. Ein verallgemeinertes definites Constraint darf links auch verallgemeinerte Membership-Ausdrücke benutzen, und rechts Komplemente von nullstelligen Konstruktoren aus Σ.

Def. Verallgemeinerte Membership-Ausdrücke haben die Form $\{x \mid \Phi(x)\}$, wobei Φ eine Formel mit der freien Variablen x ist, erzeugt von der Grammatik

$$\Phi \to t \in \text{lhs} \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \exists y \Phi \mid \forall y \Phi.$$

Dabei ist $t \in \mathsf{Tree}(\Sigma, V')$, und lhs ist eine linke Seite eines verallgemeinerten definiten Constraints.

Satz. Erfüllbarkeit von verallgemeinerten definiten Constraints ist entscheidbar (in DEXPTIME).

5.5.5 Atomic Set Constraints

Def. Ein atomisches Set Constraint hat die Form $X \subseteq Y$ oder $X \not\subseteq \emptyset$ oder $X \nparallel Y$, mit $X, Y \in \mathsf{Tree}(\Sigma, V)$ und der Interpretation

$$B(X \nparallel Y) \;=\; B(X) \cap B(Y) \neq \emptyset.$$

Satz. Erfüllbarkeit von atomischen Set-Constraint-Systemen ist entscheidbar (in kubischer Zeit).