Prof. Dr. Hans-Jörg Kreowski 31. Mai 2010

Formale Sprachen: Graphtransformation

+++++ Achtung: Die Schleifen an Knoten sind in den Beispielen nicht gezeichnet; man kann sie dennoch an den jeweils oben rechts angebrachten Markierungen erkennen, wobei die unsichtbare Markierung als $*$ gekennzeichnet ist. +++++

Kleines *NP*-Projekt mit 2. Aufgabenblatt

NP ist die berühmte und berüchtigte Klasse von Entscheidungsproblemen, die sich nichtdeterministisch in polynomieller Zeit lösen lassen. Das graphtransformatorische Pendant NP_{GT} wird in diesem kleinen Projekt untersucht.

Eine Graphtransformationseinheit $gtu = (I, P, C, T)$ ist *polynomiell*, wenn es ein Polynom p so gibt, dass für alle Ableitungen $G = G_0 \underset{r_1}{\Longrightarrow} \cdots \underset{r_n}{\Longrightarrow} G_n = H$ mit $G \in SEM(I), r_1, \cdots, r_n \in P$ und $r_1 \cdots r_n s \in L(C)$ für ein $s \in P^*$ $n \leq p(size(G))$ gilt.

Damit ist nach den Überlegungen zum Aufwand insbesondere jedes Element der Ableitungssemantik[1], bei dem also zusätzlich $s = \lambda$ und $H \in SEM(T)$ gilt, in polynomieller Zeit konstruierbar.

Eine polynomielle Graphtransformationseinheit löst das Entscheidungsproblem $D(gtu) : SEM(I) \to BOOL$ mit $D(gtu)(G) = TRUE$ gdw. $G \underset{P}{\overset{*}{\Longrightarrow}} H \in SEM(gtu)$ existiert.

Die Klasse NP_{GT} enthält alle diese Entscheidungsprobleme, unter denen sich beispielsweise die folgenden befinden:

1. das Hamiltonsche-Wege-Problem, das durch die folgende Graphtrans-

[1] bezeichnet mit $SEM(gtu)$

formationseinheit modelliert wird:

$$
\begin{array}{ll}
hp & \\
\quad initial: & unlabelled\ \&\ looped \\
\quad rules: & begin, run, end \\
\quad cond: & begin; run^*; end \\
\quad terminal: & forbidden(\bullet^*)
\end{array}
$$

hp ist polynomiell, weil die Anwendung von *begin* am Anfang und jede Anwendung von *run* eine $*$-Schleife beseitigt (und keine Regelanwendung solche Schleifen erzeugt) und *end* ohnehin nur einmal angewendet wird. Die zulässigen Ableitungen sind also um höchstens eins länger als die Zahl der $*$-Schleifen in den initialen Graphen. Die Anwedung von *end* löscht die *run*-Schleife, die von *begin* erzeugt und von den *run*-Anwendungen jeweils längs einer Kante verschoben wurde. *hp* baut somit einen einfachen Weg auf, der alle Knoten enthält, weil anfangs jeder Knoten genau eine $*$-Schleife trägt, die von *begin* und *run* gelöscht werden, während die Knoten in den Weg integriert werden. Zum Schluss ist aber keine $*$-Schleife übrig. Solche einfachen Wege, die alle Knoten besuchen, werden Hamiltonsch genannt, so dass *hp* das Hamiltonsche-Wege-Problem löst:

> $HP(G) = D(hp)(G) = TRUE$ gdw. G einen Hamiltonschen Weg besitzt.

2. das Färbungsproblem mit k Farben, das modelliert ist durch:

$$
\begin{array}{lll}
k - color & & (k \in \mathbb{N} \text{ fest}) \\
\quad initial: & unlabelled\ \&\ looped & \\
\quad rules: & \bullet^* \supseteq \bullet \subseteq \bullet^i \quad \text{für} \quad i \in [k] = \{1, \cdots, k\} & \\
\quad terminal: & forbidden(\bullet^*)\ \& & \\
 & forbidden(\bullet^i \to \bullet^i \mid i \in [k]) &
\end{array}
$$

$k - color$ ist polynomiell, weil jede Regelanwendung eine $*$-Schleife löscht, aber keine erzeugt. Eine zulässige Ableitung gibt jedem Knoten

eine „Farbe“ i, so dass keine zwei direkten Nachbarn gleich gefärbt sind. Wenn ein initialer Graph eine Färbung $c\colon V_G \to [k]$ besitzt mit $c(s_G(e)) \neq c(t_G(e))$ für alle $e \in E_G$ mit $s_G(e) \neq t_G(e)$, dann kann für jeden Knoten v gerade die Farbe $c(v)$ durch Regelanwendung gesetzt werden. $k - \mathit{color}$ löst also das Färbungsproblem mit k Farben:

> k-$\mathit{COLOR}(G) = D(k - \mathit{color})(G) = \mathit{TRUE}$ gdw. G eine Färbung mit k Farben besitzt.

3. das Unabhängigkeitsproblem *INDEP* mit $\mathit{INDEP}(G,k) = \mathit{TRUE}$ gdw. $X \subseteq V_G$ existiert mit $\#X = k$, so dass kein $e \in E_G$ Quelle und Ziel in X hat, $s_G(e), t_G(e) \in X$ also für keine Kante gilt,

4. das Cliquenproblem *CLIQUE* mit $\mathit{CLIQUE}(G,k) = \mathit{TRUE}$ gdw. $X \subseteq V_G$ existiert mit $\#X = k$ und für alle $v, v' \in X$ mit $v \neq v'$ existiert $e \in E_G$ mit $s_G(e) = v$ und $t_G(e) = v'$,

5. das Vertex-Cover-Problem *VC* mit $\mathit{VC}(G,k) = \mathit{TRUE}$ gdw. $X \subseteq V_G$ existiert mit $\#X = k$ und für alle $e \in E_G$ gilt: $s_G(e) \in X$ oder $t_G(e) \in X$,

6. das Matching-Problem *MATCH* mit $\mathit{MATCH}(G,k) = \mathit{TRUE}$ gdw. $Y \subseteq E_G$ existiert mit $\#Y = k$ und $\#\{s_G(e), t_G(e) \mid e \in Y\} = 2k$,

7. das Lange-Wege-Problem *LP* mit $\mathit{LP}(G,k) = \mathit{TRUE}$ gdw. in G ein einfacher Weg der Länge k existiert,

8. das Teilgraphenproblem *SUB* mit $\mathit{SUB}(G,G') = \mathit{TRUE}$ gdw. ein injektiver Graphmorphismus $g\colon G \to G'$ existiert.

Damit die letzten sechs Entscheidungsprobleme von Graphtransformationseinheiten gelöst werden können, müssen die Eingabepaare als Graphen dargestellt werden. Als Graphdarstellung $gr(k)$ für $k \in \mathbb{N}$ kommt eine „Blume“ mit $k+1$ „Blättern“ infrage, bei der eine *0*-Schleife den Knoten als Zähler makiert und die Zahl der *succ*-Schleifen die dargestellte Zahl k ist.

Zwei Graphen G und G' (bzw. $gr(k)$) können durch die disjukte Vereinigung $G + G'$ zu einem Graphen werden (entsprechend $G + gr(k)$). Wenn man die Komponenten unterscheiden will, kann man die Knoten von $G + G'$ noch durch unterschiedliche Schleifen markieren. Das lässt sich folgendermaßen durch Graphklassenausdrücke wiedergeben:

- $X + gr(\mathbb{N}) \rightsquigarrow SEM(X + gr(\mathbb{N}) = \{G + gr(k) \mid G \in SEM(X), k \in \mathbb{N}\}$,
- $X + X' \rightsquigarrow SEM(X + X') = \{G + G' \mid G \in SEM(X), G' \in SEM(X')\}$,
- $looped\ x \rightsquigarrow SEM(looped\ x) = \{G \in \mathcal{G}_\Sigma \mid$ genau eine $x-$Schleife pro Knoten$\}$

Mit diesen Vereinbarungen kann *INDEP* beispielsweise folgendermaßen gelöst werden:

$$
\begin{array}{ll}
indep & \\
\quad initial: & (unlabelled\ \&\ looped) + gr(\mathbb{N}) \\
\quad rules: & \bullet \overset{*}{\quad} \circ \overset{succ}{\quad} \supseteq \bullet \quad \circ \subseteq \bullet \overset{c}{\quad} \circ \\
\quad terminal: & forbidden(\bullet \overset{c}{\rightarrow} \bullet) \ \& \\
 & forbidden(\circ \overset{succ}{\quad})
\end{array}
$$

Zur Erleichterung des Modellierens wird noch das Konzept der negativen Anwendungsbedingung eingeführt. Bei der Gelegenheit werden auch *required* und *forbidden* formal definiert:

- $NAC(r = (L \supseteq K \subseteq R), L \subseteq N)$ erlaubt die Anwendung von r auf G mit dem Ansatz $g\colon L \rightarrow G$ nur, wenn es keinen Graphmorphismus $h\colon N \rightarrow G$ gibt mit $h\,|_L = g$.
- $SEM(required(G_0)) = \{G \in \mathcal{G}_\Sigma \mid injektives\ g\colon G_0 \rightarrow G$ existiert$\}$.
- $SEM(forbidden(G_0)) = \{G \in \mathcal{G}_\Sigma \mid injektives\ g\colon G_0 \rightarrow G$ existiert nicht$\}$.

Ein wesentliches Konzept für die Untersuchung von *NP* ist die Reduktion.

Eine Reduktion des Entscheidungsproblems $D\colon SEM(I) \rightarrow BOOL$ auf das Entscheidungsproblem $D'\colon SEM(I') \rightarrow BOOL$ ist definiert als eine Abbildung $RED\colon SEM(I) \rightarrow SEM(I')$, die sich in polynomieller Zeit berechnen lässt und korrekt ist. Letzteres bedeutet:

$$D(G) = D'(RED(G)) \quad \text{für alle} \quad G \in SEM(I).$$

Man schreibt dann auch $D \leq D'$.

Eine Reduktion *RED* wird durch eine polynomielle Graphtransformationseinheit $red = (I, P, C, I')$ realisiert, falls für alle $G \in SEM(I)$ gilt:

$$G \overset{*}{\underset{P}{\Longrightarrow}} H \in SEM(red) \quad \text{gdw.} \quad H = RED(G).$$

Beispielsweise realisiert

$HP - 2 - HP_{AB}$

$initial:$ *unlabelled* & *looped*

$rules:$ $set_{AB} = (\emptyset \supseteq \emptyset \subseteq \bullet^A \quad \bullet^B)$

$connect = (\bullet^A \quad \bullet^* \quad \bullet^B \supseteq \bullet^A \quad \bullet \quad \bullet^B \subseteq \bullet^A \to \bullet^c \to \bullet^B)$

$reset = (\bullet^c \supseteq \bullet \subseteq \bullet^*)$

$cond:$ $set_{AB}; connect!; reset!$

$terminal:$ $unlabelled$ & $looped_{AB}$

eine Reduktion von *HP* auf das spezielle Hamiltonsche-Wege-Problem, bei dem noch ein Anfangsknoten $\bullet^A$ und ein Endknoten $\bullet^B$ vorgegeben sind. Der Graphklassenausdruck $looped_{AB}$ ist wie *looped* definiert, wenn man davon absieht, dass eine $*$-Schleife durch eine A-Schleife und eine andere $*$-Schleife durch eine B-Schleife ersetzt sind. Das Ausrufungszeichen in der Kontrollbedingung verschärft den Stern und verlangt, dass die Regel nicht nur beliebig oft, sondern so lange wie möglich iteriert wird. Die Regel set_{AB} wird anfangs einmal angewendet und produziert zwei neue Knoten, die durch *connect*! nacheinander mit allen anderen Knoten verbunden werden. Diese erhalten zwischendurch eine c-Schleife, die von *reset* jeweils auf unmarkiert zurückgesetzt wird.

Aufgaben

Zeige von zwei bis drei Problemen, dass sie in NP_{GT} liegen.

Beachte, dass dazu eine Graphtransformationseinheit entworfen werden muss, die polynomiell ist, was einer Begründung oder eines Beweises bedarf. Als Probleme können die im Material genannten genommen werden, soweit ihre Mitgliedschaft in NP_{GT} nicht bereits nachgewiesen ist. Aber andere Probleme aus der Literatur sind ebenfalls willkommen. Wie in den Beispielen *hp*, *k-color*, *indep* und *sat3* wäre es sinnvoll, neben der modellierten Graphtransformationseinheit auch eine logische bzw. graphentheoretische Beschreibung anzugeben und deren Übereinstimmung mit der Entscheidung der Graphtransformationseinheit zu begründen oder zu beweisen.

Realisiere außerdem zwei bis drei Reduktionen zwischen Entscheidungsproblemen mit Hilfe von Graphtransformationseinheiten.

Als Vorbild kann hier $HP - 2 - HP_{AB}$ dienen. Die Probleme können frei gewählt werden, die ineinander reduziert werden. Als Hilfe seien einige Kandidaten genannt: $HAM \leq HP_{AB}$, $HP_{AB} \leq HP$, $HP \leq LP$, $HAM \leq SUB$, $INDEP \leq CLIQUE$, $CLIQUE \leq VC$, $VC \leq INDEP$. Neben der Konstruktion ist wieder auch zu begründen oder zu beweisen, dass die Konstruktion funktioniert, also polynomiell ist und die Entscheidungen richtig überträgt.