Constraint Graphs

A **constraint graph** is a function $g \in V \to V^0 \cup V^1 \cup V^2$ such that V is a finite set of type variables and $\{\, (X, Y) \mid gX = \langle Y \rangle \,\}$ is terminating.

The letter g will always denote a constraint graph. Given a constraint graph g, a type variable X is called a

- **terminal node** if $gX = \langle\rangle$.
- **forward node** if $gX \in V^1$.
- **branching node** if $gX \in V^2$.

$$\begin{array}{rcll} \mathit{Fwd}\ g & \overset{\text{def}}{=} & \{\, (X, Y) \mid gX = \langle Y \rangle \,\} & \text{forward edges} \\ \mathit{Edge}\ g & \overset{\text{def}}{=} & \mathit{Fwd}\ g \cup \{\, (X, Y_i) \mid gX = \langle Y_1, Y_2 \rangle \wedge i \in \{1, 2\} \,\} & \text{edges} \\ \mathit{Con}\ g & \overset{\text{def}}{=} & \mathit{Fwd}\ g \cup \{\, (X, Y_1 \to Y_2) \mid gX = \langle Y_1, Y_2 \rangle \,\} & \text{constraint} \end{array}$$

Note that *Fwd g* is a confluent and terminating relation. We define:

$$\hat{g}X \overset{\text{def}}{=} \text{normal form of } X \text{ with respect to } \mathit{Fwd}\ g$$

A constraint graph g is **acyclic** if *Edge g* is terminating. If g is acyclic, every node of g represents a type. We define the corresponding function $\mathit{ty}\ g \in \mathit{Dom}\ g \to \mathit{Ty}$ by recursion on *Edge g*:

$$\begin{array}{lll} \mathit{ty}\ g\ X = X & \text{if} & gX = \langle\rangle \\ \mathit{ty}\ g\ X = \mathit{ty}\ g\ Y & \text{if} & gX = \langle Y \rangle \\ \mathit{ty}\ g\ X = \mathit{ty}\ g\ Y_1 \to \mathit{ty}\ g\ Y_2 & \text{if} & gX = \langle Y_1, Y_2 \rangle \end{array}$$

Proposition 1 If g is acyclic, then $\{\, (X, \mathit{ty}\ g\ X) \mid X \in \mathit{Dom}\ g \wedge gX \neq X \,\}$ is a solved constraint equivalent to *Con g*.

Proposition 2 *Con g* satisfiable $\iff$ g acyclic.

A **flat constraint** is a constraint C such that for every pair $(T_1, T_2) \in C$ both components T_1 and T_2 are type variables. A **combined constraint** is a pair (g, E) consisting of a graph function g and a flat constraint E such that E contains only variables in *Dom g*. **Equivalence** of combined constraints is defined as follows:

$$(g, E) \approx (g', E') \quad \overset{\text{def}}{\iff} \quad \mathit{Con}\ g \cup E \approx \mathit{Con}\ g' \cup E' \ \wedge\ \mathit{Dom}\ g = \mathit{Dom}\ g'$$

With the unification rules shown in the following proposition one can compute for (g, E) a graph function g' such that $(g, E) \approx (g', \emptyset)$.

Proposition 3 (Unification Rules)

1. $(g,\ E \cup \{X = Y\}) \ \approx\ (g, E)$
   if $\hat{g}X = \hat{g}Y$ (Reflexivity)
2. $(g,\ E \cup \{X = Y\}) \ \approx\ (g[\hat{g}X := \hat{g}Y],\ E \cup \{X_1 = Y_1,\ X_2 = Y_2\})$
   if $\hat{g}X \neq \hat{g}Y,\ g(\hat{g}X) = \langle X_1, X_2 \rangle$ and $g(\hat{g}Y) = \langle Y_1, Y_2 \rangle$ (Decomposition)
3. $(g,\ E \cup \{X = Y\}) \ \approx\ (g[\hat{g}X := \hat{g}Y],\ E)$
   if $\hat{g}X \neq \hat{g}Y$ and $g(\hat{g}X) = \langle\rangle$ (Replacement)
4. $(g,\ E \cup \{X = Y\}) \ \approx\ (g[\hat{g}Y := \hat{g}X],\ E)$
   if $\hat{g}X \neq \hat{g}Y$ and $g(\hat{g}Y) = \langle\rangle$ (Replacement)

All rules leave the number of nodes unchanged, and Rules 2–4 increase the number of forward nodes. Hence one arrives at a combined constraint $(g', \emptyset)$ after $O(|E| + |Dom\ g|)$ steps if one begins with (g, E). Since $\hat{g}$ can be computed in $O(|Dom\ g|)$, a combined constraint (g, E) can be simplified in time $O(|E| + |Dom\ g|^2) = O(|Dom\ g|^2)$. Since g can be checked for cycles in linear time, a combined constraint can be solved in quadratic time (i.e., $O(|Dom\, g|^2)$).

Rather than checking the acyclicity of the constraint graph at the end, one can check the acyclicity at the beginning and strengthen the decomposition and replacement rules such that they don't introduce cycles.