Syntax	Semantics	$\mathcal{FL}_0$	$\mathcal{FL}_\bot$	$\mathcal{FL}_\neg$	$\mathcal{ALN}$
$\top$	$\Delta^\mathcal{I}$	x	x	x	x
$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$	x	x	x	x
$\forall r.C$	$\{x \in \Delta^\mathcal{I} \mid \forall y : (x, y) \in r^\mathcal{I} \to y \in C^\mathcal{I}\}$	x	x	x	x
$\bot$	$\emptyset$		x	x	x
$\neg P, P \in N_C$	$\Delta^\mathcal{I} \setminus P^\mathcal{I}$			x	x
$(\geq n\ r), n \in \mathbb{N}$	$\{x \in \Delta^\mathcal{I} \mid \#\{y \mid (x, y) \in r^\mathcal{I}\} \geq n\}$				x
$(\leq n\ r), n \in \mathbb{N}$	$\{x \in \Delta^\mathcal{I} \mid \#\{y \mid (x, y) \in r^\mathcal{I}\} \leq n\}$				x

Table 1: Syntax and semantics of concept descriptions.

is a finite set of side conditions. If the set S contains only subsumption conditions, then M is called matching problem under subsumption conditions. *The substitution σ is a* solution *(*matcher*) of M iff it is a matcher of $C \equiv^? D$ that satisfies every side condition in S.*

In the next section, we will restrict the attention to matching problems under subsumption conditions. Section 4 then treats matching problems under acyclic side conditions.

In order to define matching problems under acyclic side conditions, we say that a variable X *directly depends* on a variable Y in S iff S contains a side condition $X\ \rho\ E$ such that Y occurs in E. If there are $n \geq 1$ variables $X_1, \ldots, X_n$ such that X_i directly depends on X_{i+1} in S $(1 \leq i \leq n-1)$, then we say that X_1 *depends* on X_n in S. The set of side conditions S is *cyclic* iff there is a variable X that depends on itself in S; otherwise, S is *acyclic*.

3 Matching under subsumption conditions

Let $\mathcal{L}$ be one of the DLs $\mathcal{FL}_\bot, \mathcal{FL}_\neg, \mathcal{ALN}$. We present a polynomial time algorithm that, given an $\mathcal{L}$-matching problems under subsumption conditions, returns a least matcher (w.r.t. the ordering $\sqsubseteq$ on substitutions) if the problem is solvable, and "fail" otherwise. In principle, the algorithm iterates the application of $\text{MATCH}_\mathcal{L}$ until a fixpoint is reached. However, the matcher computed in one step is used to modify the matching problem to be solved in the next step. Given an $\mathcal{L}$-matching problem under subsumption conditions $M := \langle C \equiv^? D, S\rangle$ and a substitution σ, we define

$$M_\sigma := \{C \equiv^? D\} \cup \{\sigma(X) \sqsubseteq^? E \mid X \sqsubseteq^? E \in S\}.$$

Recall that $\sigma(X) \sqsubseteq^? E$ abbreviates the matching problem $\sigma(X) \equiv^? \sigma(X) \sqcap E$. Thus M_σ is a system of $\mathcal{L}$-matching problems without side conditions, to which $\text{MATCH}_\mathcal{L}$ can be applied.

Algorithm 4 *Let $M := \langle C \equiv^? D, S\rangle$ be an $\mathcal{L}$-matching problem under subsumption conditions. Then, the algorithm $\text{MATCH}^\sqsubseteq_\mathcal{L}(M)$ works as follows:*

1. *$\sigma(X) := \bot$ for all variables X;*
2. *If $\text{MATCH}_\mathcal{L}(M_\sigma)$ returns "fail", then return "fail";*
   else if $\sigma \equiv \text{MATCH}_\mathcal{L}(M_\sigma)$, then return σ;
   else $\sigma := \text{MATCH}_\mathcal{L}(M_\sigma)$; continue with 2.

Let σ_0 denote the substitution defined in step 1 of the algorithm, and σ_t $(t \geq 1)$ the matcher computed in the t-th iteration of Step 2. Note that σ_t is undefined if $\text{MATCH}_\mathcal{L}$ returns "fail" in the t-th iteration or if the algorithm has stopped before the t-th iteration.

To show that the algorithm is correct, we must show soundness, completeness, and termination, i.e., i) if the algorithm terminates and returns a substitution, then this substitution in fact solves the problem; ii) if the algorithm terminates and returns "fail", then there indeed is no solution; and iii) the algorithm halts on every input. The following lemma proves soundness and completeness of the algorithm. The first two items establish a loop invariant.

Lemma 5 *Let $M := \langle C \equiv^? D, S\rangle$ be an $\mathcal{L}$-matching problem under subsumption conditions.*

1. *If σ_t is defined and τ is a solution of M, then $\sigma_t \sqsubseteq \tau$.*
2. *If σ_t, σ_{t+1} are defined, then $\sigma_t \sqsubseteq \sigma_{t+1}$.*
3. *If $\text{MATCH}^\sqsubseteq_\mathcal{L}(M)$ returns the substitution σ, then σ solves M (soundness).*
4. *If $\text{MATCH}^\sqsubseteq_\mathcal{L}(M)$ returns "fail", then M has no solution (completeness).*

PROOF. 1. Obviously, the claim is true for σ_0. Assume that $\sigma_t \sqsubseteq \tau$, and that σ_{t+1} is defined. To prove $\sigma_{t+1} \sqsubseteq \tau$, it is sufficient to show that τ solves M_{σ_t} since σ_{t+1} is the least solution of M_{σ_t}. Since τ solves M, we know that it solves $C \equiv^? D$ and that $\tau(X) \sqsubseteq \tau(E)$ for all $X \sqsubseteq^? E \in S$. The induction assumption $\sigma_t \sqsubseteq \tau$ implies $\sigma_t(X) \sqsubseteq \tau(X)$, and thus $\sigma_t(X) \sqsubseteq \tau(E)$, which shows that τ solves M_{σ_t}.

2. Obviously, $\sigma_0 \sqsubseteq \sigma_1$. Now assume that $\sigma_{t-1} \sqsubseteq \sigma_t$. Together with the fact that σ_t solves $M_{\sigma_{t-1}}$, this implies that σ_{t+1} solves the system $M_{\sigma_{t-1}}$. Since σ_t is the least solution of $M_{\sigma_{t-1}}$, we can conclude $\sigma_t \sqsubseteq \sigma_{t+1}$.

3. Assume that $\sigma = \sigma_t$. By definition of $\text{MATCH}^\sqsubseteq_\mathcal{L}$, $C \equiv \sigma_t(D)$. It remains to show that σ_t solves the side conditions. We know that $\sigma_t \equiv \sigma_{t+1}$ and σ_{t+1} solves M_{σ_t}. Thus, $\sigma_t(X) \sqsubseteq \sigma_{t+1}(E) \equiv \sigma_t(E)$ for every $X \sqsubseteq^? E \in S$.

4. Assume that $\text{MATCH}^\sqsubseteq_\mathcal{L}(M)$ returns "fail," and that σ_t is the last substitution computed by the algorithm. Now assume that τ solves M. As in the proof of 1. we can show that τ solves M_{σ_t}. Consequently, M_{σ_t} is solvable, and thus $\text{MATCH}(M_{\sigma_t})$ returns the least matcher of this system, in contradiction to the assumption that $\text{MATCH}^\sqsubseteq_\mathcal{L}(M)$ returns "fail" in this step of the iteration.

■