A A DLP Program Simulating a Turing Machine

We next show how a Turing Machine can be encoded by a suitable DLP program simulating its computation. Let M be a Turing Machine given by the 4-uple $\langle K, \Sigma, \delta, s_0 \rangle$, where K is a finite set of states, $s_0 \in K$ is the initial state, Σ is a finite set of symbols constituting the alphabet (with $\sqcup \notin \Sigma$ standing for the blank symbol), and $\delta : K \times \Sigma \rightarrow K \times \Sigma \times \{l, r, \lambda\}$ is the transition function describing the behavior of the machine. Given the current state and the current symbol, δ specifies the next state, the symbol to be overwritten on the current one, and the direction in which the cursor will move on the tape (l, r, λ standing for left, right, stay, respectively). Besides the initial state, there is another special state, which is called final state; the machine halts if the machine reaches this state at some point. Each configuration of M can be encoded in a program P_M by means of the following predicates.

- $tape(P, Sym, T)$: the tape position P stores the symbol Sym at time step T. For each time step, there is an instance of such predicate for every actually used position of the tape.
- $position(P, T)$: the head of M reads the position P on tape at time step T. $position$ has a single true ground instance for each time step.
- $state(St, T)$: at time step T M is in the state St. $state$ has a single true ground instance for each time step.

P_M encodes the transition function δ in the following way: For each St_c, Sym_c, St_n, Sym_n, D, such that $\delta(St_c, Sym_c) = (St_n, Sym_n, D)$ we add to P_M a fact of the form $delta(St_c, Sym_c, St_n, Sym_n, D)$. The initial input is encoded by a proper set of facts describing all tape positions at the first time step (facts of the form $tape(P, Sym, 0)$), a fact of the form $state(s_0, 0)$, and a fact of the form $position(P, 0)$ where P is the initial position of the head. The rules defining the evolution of the machine configurations are reported next. For the sake of readability, we exploit some comparison built-ins, that could be easily simulated by means of suitable predicates.

(r_1)	$position(P, s(T))$	$\text{:-}\ position(s(P), T), state(St, T), tape(s(P), Sym, T), delta(St, Sym, \_, \_, l).$
(r_2)	$position(s(P), s(T))$	$\text{:-}\ position(P, T), state(St, T), tape(P, Sym, T), delta(St, Sym, \_, \_, r).$
(r_3)	$position(P, s(T))$	$\text{:-}\ position(P, T), state(St, T), tape(P, Sym, T), delta(St, Sym, \_, \_, \lambda).$
(r_4)	$state(St1, s(T))$	$\text{:-}\ position(P, T), state(St, T), tape(P, Sym, T), delta(St, Sym, St1, \_, \_).$
(r_5)	$tape(P, Sym1, s(T))$	$\text{:-}\ position(P, T), state(St, T), tape(P, Sym, T), delta(St, Sym, \_, Sym1, \_).$
(r_6)	$tape(P, Sym, s(T))$	$\text{:-}\ position(P1, T), tape(P, Sym, T), P \neq P1.$
(r_7)	$tape(P, \sqcup, T)$	$\text{:-}\ position(P, T), lastUsedPos(L, T), P > L.$
(r_8)	$lastUsedPos(L, s(T))$	$\text{:-}\ lastUsedPos(L, T), position(P, T), P \leq L.$
(r_9)	$lastUsedPos(P, s(T))$	$\text{:-}\ lastUsedPos(L, T), position(P, T), P > L.$

First three rules encode how the tape position changes according to the transition function; the fourth updates the state. Rule r_5 updates, for each time step, the current tape position with the new symbol to be stored, with rule r_6 stating that all other positions remain unchanged. Rules r_7, r_8, r_9 allow to manage the semi-infinite tape. Indeed, the whole tape is not explicitly encoded; rather, each tape position is initialized with a blank symbol when reached for the first time (moving right, the tape being limited at left).

Given a valid tape x encoded by means of a set X of facts of the form $tape(p, s, 0)$, one can show that the computation of $(P_M \cup X)^\gamma$ follows in one-to-one correspondence the computation of M on the tape x. γ is unique and contains a single component C having a corresponding module M. We have that $S_0 = EDB(P_M)$, and $S_1 = S_0 \cup \Phi^{\infty}_{M,S_0}(\emptyset)$. Let $\Phi(t) = \Phi^{t}_{M,S_0}(\emptyset)$. Then, the value of $\Phi(t)$ directly corresponds to the step t of M. It is easy to note that, at step $t + 1$, $\Phi(t + 1)$ can be larger than $\Phi(t)$ only if, at step t, $\Phi(t)$ contains an atom $state(st, t)$ for st not a final state. In such a case by means of rules r_1 through r_5, new atoms of form $position(p, sym, t + 1), state(st, t + 1), tape(p, sym, t + 1)$ are added to $\Phi(t + 1)$.