Encoding of a System Description

The encoding $\Pi(\mathcal{SD})$ of system description $\mathcal{SD}$ consists of the encoding of the signature of $\mathcal{SD}$ and rules obtained from statements of $\mathcal{SD}$.

- **Encoding of the Signature**

  We start with the encoding $\text{sig}(\mathcal{SD})$ of the signature of $\mathcal{SD}$.

  - For each constant symbol $c$ which has a sort $\text{sort\_name}$ other than fluent, static or action, $\text{sig}(\mathcal{SD})$ contains $\text{sort\_name}(c)$ (1)

  - For every static $g$ of $\mathcal{SD}$, $\text{sig}(\mathcal{SD})$ contains $\text{static\_name}(g)$ (2)

  - For every inertial fluent $f$ of $\mathcal{SD}$, $\text{sig}(\mathcal{SD})$ contains $\text{fluent}(\text{inertial}, f)$ (3)

  - For every defined fluent $f$ of $\mathcal{SD}$, $\text{sig}(\mathcal{SD})$ contains $\text{fluent}(\text{defined}, f)$ (4)

  - For every action $a$ of $\mathcal{SD}$, $\text{sig}(\mathcal{SD})$ contains $\text{action}(a)$ (5)

- **Encoding of Statements of $\mathcal{SD}$**

  For this encoding we only need two steps, 0 and 1, which stand for the beginning and the end of a transition. This is sufficient for describing a single transition; however, later, we describe longer chains of events and let steps range over $[0,n]$ for some constant $n$. To allow an easier generalization of the program we encode steps by using constant $n$ for the maximum number of steps, as follows:

  $\#\text{const\_n} = 1$.

  $\text{step}(0..n)$. (7)

  As in our blocks-world example, we introduce a relation $\text{holds}(f,i)$ which says that fluent $f$ is true at step $i$. To simplify the description of the encoding, we also introduce a new notation, $h(l,i)$ where $l$ is a domain literal and $i$ is a step. If $f$ is a fluent then by $h(l,i)$ we denote $\text{holds}(f,i)$ if $l = f$ or $\neg\text{holds}(f,i)$ if $l = \neg f$. If $l$ is a static literal then $h(l,i)$ is simply $l$. We also need relation $\text{occurs}(a,i)$ which says that action $a$ occurred at step $i$; $\text{occurs}([a_0,\ldots,a_k],i) = \text{def} \{ \text{occurs}(a_j,i) : 0 \leq j \leq k \}$.

  We use this notation to encode statements of $\mathcal{SD}$ as follows:
– For every causal law

\[ a \text{ causes } l \text{ if } p_0, \ldots, p_m \]

\( \Pi(\mathcal{D}) \) contains

\[ h(l, I+1) \leftarrow h(p_0, I), \ldots, h(p_m, I), \]
\[ \text{occurs}(a, I), \]
\[ I < n. \]

(8)

– For every state constraint

\[ l \text{ if } p_0, \ldots, p_m \]

\( \Pi(\mathcal{D}) \) contains

\[ h(l, I) \leftarrow h(p_0, I), \ldots, h(p_m, I). \]

(9)

– \( \Pi(\mathcal{D}) \) contains the CWA for defined fluents:

\[ \neg \text{holds}(F, I) \leftarrow \text{fluent}(\text{defined}, F), \]
\[ \text{not holds}(F, I). \]

(10)

– For every executability condition

\[ \text{impossible } a_0, \ldots, a_k \text{ if } p_0, \ldots, p_m \]

\( \Pi(\mathcal{D}) \) contains

\[ \neg \text{occurs}(a_0, I) \text{ or } \ldots \text{ or } \neg \text{occurs}(a_k, I) \leftarrow h(p_0, I), \ldots, h(p_m, I). \]

(11)

– \( \Pi(\mathcal{D}) \) contains the Inertia Axiom:

\[ \text{holds}(F, I+1) \leftrightarrow \text{fluent}(\text{inertial}, F), \]
\[ \text{holds}(F, I), \]
\[ \neg \text{not holds}(F, I+1), \]
\[ I < n. \]

(12)

\[ \neg \text{holds}(F, I+1) \leftrightarrow \text{fluent}(\text{inertial}, F), \]
\[ \neg \text{holds}(F, I), \]
\[ \text{not holds}(F, I+1), \]
\[ I < n. \]

(13)

– \( \Pi(\mathcal{D}) \) contains CWA for actions:

\[ \neg \text{occurs}(A, I) \leftarrow \text{not occurs}(A, I). \]

(14)