Errata & additions to

Applying learner modelling for user interface assistance in simulative training systems

Alexander Hörnlein, Frank Puppe

Dept. for Artificial Intelligence and Applied Computer Science, University of Würzburg
{hoernlein, puppe}@informatik.uni-wuerzburg.de

<u>The objective function (p. 5) is correctly defined the following way:</u>

We call T the set of all tasks of the presented case. For each task t there is a set $Q_{0,t}$ of states where this task is reasonable. We express that with the objective function $o: T \times Q \to \mathcal{P}(Q),\ o(t,q) = \begin{cases} Q_{F,t},\ if\ q \in Q_{0,t} \\ \varnothing\ otherwise \end{cases}$. $\mathcal{P}(Q)$ is the power set of Q, $Q_{F,t}$ is the set of all final states for t.

<u>The following is a formally more correct definition of the overlay model (p.7):</u>

We denote with C and S two finite sets: C is the set of concepts, S is a set of symbols. $I = [n_1, n_2]$ is an interval with $n_1 \in \mathbb{N}^-, n_2 \in \mathbb{N}^+$. We call $m_1 : S \to I$ the *symbolic value function*. For $a, b \in S$ (as m_1 is injective), we define $a \le b$ if $m_1(a) \le m_1(b)$, so we have a natural order over S. Further, S contains an element s_0 with $m_1(s_0) = 0$.

We define the surjective function $m_2 : I \to S$ which we call the *inverted symbolic value function* and the injective function $m_3 : C \to I$ which we call the *numerical score function*.

We define an overlay model M as the 6-tuple $M = (C, S, I, m_1, m_2, m_3)$.

To model the way how we change the overlay model, we define a function $g : \mathfrak{F}(C, I) \times C \times S \rightarrow \mathfrak{F}(C, I)$, where $\mathfrak{F}(C, I)$ is the set of all function $f : C \rightarrow I$ with:

$$m_3' := g\left(m_3, c, s\right) \textit{ with } m_3'\left(c_i\right) = \begin{cases} \min\left(m_3\left(c_i\right) + m_1\left(s\right), n_2\right), \textit{if } c_i = c \textit{ and } m_1\left(s\right) > 0 \\ \max\left(m_3\left(c_i\right) + m_1\left(s\right), n_1\right), \textit{if } c_i = c \textit{ and } m_1\left(s\right) < 0 \\ m_3\left(c_i\right), \textit{otherwise} \end{cases}$$

Example:

- Set of concepts $C = \left\{\text{"}\textit{task 'diagnose' consists of subtask 'switch to diagnose', ...}\text{"}, ...\right\}$
- Set of symbols $S = \left\{N2, N1, P0, P1, P2\right\}$
- Interval $I = \left[-15, 15\right]$
- m_1 is defined with:

s	$N2$	$N1$	$P0$	$P1$	$P2$
$m_1(s)$	-10	-5	0	5	10

- m_2 is defined with:

$$m_2\left(n\right) = \begin{cases} N2, \textit{if } n \in \left[-15, 10\right] \\ N1, \textit{if } n \in \left(-10, -5\right] \\ P0, \textit{if } n \in \left(-5, 5\right) \\ P1, \textit{if } n \in \left[5, 10\right) \\ P2, \textit{if } n \in \left[10, 15\right] \end{cases}$$

- $m_3\left(c\right) = 0$ for all $c \in C$

If we apply a rule like IF … THEN APPLY $g\left(m_3, c_1, N2\right)$, then we get M with

$$m_3' := g\left(m_3, c_1, N2\right) \textit{ with } m_3'\left(c_i\right) = \begin{cases} -10, \textit{if } c_i = c_1 \\ 0, \textit{otherwise} \end{cases}.$$

Another application of the same rule would yield

$$m_3'' := g\left(m_3', c_1, N2\right) \textit{ with } m_3''\left(c_i\right) = \begin{cases} -15, \textit{if } c_i = c_1 \\ 0, \textit{otherwise} \end{cases}.$$