RULE NORMALIZATION

In this section, we present the normalization function, NORM, which converts an original SLEEC rule to a set of normalized SLEEC DSL rules. The original SLEEC DSL rule follows the syntax: **when** $e \wedge p$ **then** *resp* where e is an event symbol, p is a proposition and *resp* is a response. A response is one of the following:

(1) **not** e **within** t
(2) e **within** t
(3) e **within** t **otherwise** *resp*
(4) $resp_1$ **unless** *ptextbf(then* $resp_2$)? where the expression $(*)?$ indicates $*$ is optional.

Let an original SLEEC DSL rule "r_o = **when** $(e \wedge p)$ **then** *response*" be given. The result of normalizing r_o is the set of normalized rules $= \{\textbf{when}\ e\ \textbf{then} \bigvee_{cob} \mid \bigvee_{cob} \in \textsc{Norm}(resp, p)\}$ where the normalization function NORM is defined in Fig. 7. Given a response *resp*, NORM flattens *resp* into a set of obligation chains by traversing the nested structure of *resp* top-down, and then merges the normalization results bottom-up. Note that in a nested response, a chain of *unless* is left associative (e.g., *A unless B unless C* is equivalent as ((*A unless B*) *unless C*)), and *otherwise* has a higher precedence than *unless* by default (e.g., *A unless B otherwise C* is equivalent to *A unless* (*B otherwise C*). NORM also records and recursively distributes the triggering condition p to each case. The set of obligation chains returned by NORM can then be turned into a set of normalized rules by disturbing the triggering event e to them.

COROLLARY 1. *For any original SLEEC rule with n syntax tokens, the size of the normalized SLEEC rule is $O(n)$*

Example 20. Consider the original SLEEC rule r_o shown in Fig. 8. Applying the normalization function NORM on r_o yields two normalized rules r_{n1} and r_{n2} shown in Fig. 8.

The semantics of the normalized SLEEC DSL is shown in Fig. 9.

$$
\textsc{Norm}(resp, p) = \begin{cases}
\{p \Rightarrow e\ \textbf{within}\ t\} & \textit{if } resp = e\ \textbf{within}\ t \\
\{p \Rightarrow \textbf{not}\ e\ \textbf{within}\ t\} & \textit{if } resp = \textbf{not}\ e\ \textbf{within}\ t \\
\textsc{Norm}(resp_1, p\ \textbf{and not}\ p') \cup \textsc{Norm}(resp_2, p\ \textbf{and}\ p') & \textit{if } resp = resp_1\ \textbf{unless}\ p'\ \textbf{then}\ resp_2 \\
\textsc{Norm}(resp_1, p\ \textbf{and not}\ p') & \textit{if } resp = resp_1\ \textbf{unless}\ p' \\
\{\textsc{Norm}(e\ \textbf{within}\ t, p)\ \textbf{otherwise}\ \bigvee_{cob} \mid \bigvee_{cob} \in \textsc{Norm}(resp_2, \top)\} & \textit{if } r_{op} = e\ \textbf{within}\ t\ \textbf{otherwise}\ resp_2
\end{cases}
$$

Figure 7: Function NORM takes *resp* and p and returns a set of normalized pseudo-rules.

$$
\begin{array}{c}
\text{Original SLEEC Rule} \\
r_o = \textbf{when}\ e_1\ \textbf{and}\ p_1\ \textbf{then}\ e_2\ \textbf{within}\ t_1\ \textbf{otherwise}\ (e_3\ \textbf{within}\ t_2\ \textbf{unless}\ p_3\ \textbf{then}\ e_4\ \textbf{within}\ t_3) \\
\hline
\text{Normalized SLEEC Rules} \\
r_{n1} = \textbf{when}\ e_1\ \textbf{then}\ (p_1 \Rightarrow (e_2\ \textbf{within}\ t_1))\ \textbf{otherwise}\ (\textbf{not}\ p_3 \Rightarrow e_3\ \textbf{within}\ t_2) \\
r_{n2} = \textbf{when}\ e_1\ \textbf{then}\ (p_1 \Rightarrow (e_2\ \textbf{within}\ t_1))\ \textbf{otherwise}\ (p_3 \Rightarrow e_4\ \textbf{within}\ t_3)
\end{array}
$$

Figure 8: An example of SLEEC Rule normalization. Given an original SLEEC rule r_o, applying function NORM yields two normalized rules r_{n1} and r_{n2}.

$$
\begin{array}{lll}
\sigma \models_i p & \text{iff} & \mathbb{M}_i(p) \\
\sigma \models_i e\ \textbf{within}\ t & \text{iff} & \exists j \in [i,n].(e \in \mathcal{E}_j \wedge \delta_j \in [\delta_i, \delta_i + \mathbb{M}_j(t)]) \\
\sigma \not\models_i^j e\ \textbf{within}\ t & \text{iff} & \delta_j = \delta_i + \mathbb{M}_i(t) \wedge \forall j' \in [i,j](e \notin \mathcal{E}_{j'}) \\
\sigma \models_i \textbf{not}\ e\ \textbf{within}\ t & \text{iff} & \exists j(\sigma \not\models_i^j e\ \textbf{within}\ t) \\
\sigma \not\models_i^j \textbf{not}\ e\ \textbf{within}\ t & \text{iff} & \sigma \models e\ \textbf{within}\ t \wedge \forall j' \in [i,j)(\sigma \not\models_i^j e\ \textbf{within}\ t) \\
\sigma \models_i (p \Rightarrow ob) & \text{iff} & \sigma \models_i p \Rightarrow \sigma \models_i ob \\
\sigma \not\models_i^j (p \Rightarrow ob) & \text{iff} & \sigma \models_i \textbf{not}\ p \wedge \sigma \not\models_i^j ob \\
\sigma \models_i cob^+\ \textbf{otherwise}\ \bigvee_{cob} & \text{iff} & \sigma \models_i cob^+ \vee \exists j(\sigma \not\models_i^j cob^+ \wedge \sigma \models_j \bigvee_{cob}) \\
\sigma \not\models_i^j cob^+\ \textbf{otherwise}\ \bigvee_{cob} & \text{iff} & \exists j' \in [i,j](\sigma \not\models_i^{j'} cob^+ \wedge \sigma \not\models_{j'}^j \bigvee_{cob}) \\
\sigma \models \textbf{when}\ e\ \textbf{and}\ p\ \textbf{then}\ \bigvee_{cob} & \text{iff} & \forall i \in [1,n]((e \in \mathcal{E}_i \wedge \mathbb{M}_i(p)) \Rightarrow \sigma \models_i \bigvee_{cob}) \\
\hline
\sigma \models_i \textbf{not} \bigvee_{cob} & \text{iff} & \exists j(\sigma \not\models_i^j \bigvee_{cob}) \\
\sigma \models \textbf{exists}\ e\ \textbf{and}\ p\ \textbf{while}\ \bigvee_{cob} & \text{iff} & \exists i \in [1,n](e \in \mathcal{E}_i \wedge \mathbb{M}_i(p) \wedge \sigma \models_i \bigvee_{cob}) \\
\sigma \models \textbf{exists}\ e\ \textbf{and}\ p\ \textbf{while not} \bigvee_{cob} & \text{iff} & \exists i \in [1,n](e \in \mathcal{E}_i \wedge \mathbb{M}_i(p) \wedge \sigma \models_i \textbf{not} \bigvee_{cob})
\end{array}
$$

Figure 9: Semantics of normalized SLEEC DSL defined over trace $\sigma = (\mathcal{E}_1, \mathbb{M}_1, \delta_1) \dots (\mathcal{E}_n, \mathbb{M}_n, \delta_n)$.