Supplementary Material for "Analyzing and Debugging Normative Requirements via Satisfiability Checking"

Anonymous Author(s)

ACM Reference Format:
Anonymous Author(s). 2023. Supplementary Material for "Analyzing and Debugging Normative Requirements via Satisfiability Checking" . In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 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 = {**when** e **then** $\bigvee_{cob} \mid \bigvee_{cob} \in$ Norm$(resp, p)$} where the normalization function Norm is defined in Fig. 1. 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 1. Consider the original SLEEC rule r_o shown in Fig. 2. Applying the normalization function Norm on r_o yields two normalized rules r_{n1} and r_{n2} shown in Fig. 2.

The semantics of the normalized SLEEC DSL is shown in Fig. 3.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 SITUATIONAL CONFLICT

A SLEEC DSL rule may interact with other rules in a contradictory manner under specific situations, which can lead to conflicts between different rules. If the conflicting situations are feasible, then the stakeholders need to resolve the conflict. We define the notion of *situational conflict* as a generalization of a vacuous conflict.

Definition 1 (Situation)**.** For a rule r = **when** e **and** p **then** $\bigvee_{cob}$, an *r-triggering situation* is a tuple $(\sigma_0^k, \vec{M}_k)$ where $\sigma_0^k = (\mathcal{E}_0, \mathbb{M}_0, \delta_0)$ … $(\mathcal{E}_k, \mathbb{M}_k, \delta_k)$ is the prefix of a trace up to and including state k, $\vec{M}_k = \mathbb{M}_k \ldots \mathbb{M}_n$ are *partial* measures (i.e., the functions $\mathbb{M}_n \in \vec{M}_k$ might be undefined for some measures $m \in M$) for states from $k+1$ onwards, and σ_0^k triggers r in its kth state (i.e., $e \in \mathcal{E}_k \wedge \mathbb{M}_k(p) = \top$).

Intuitively, an r-triggering situation $(\sigma_0^k, \vec{M}_k)$ consists of (1) the complete state information of the past (until r is triggered) where all events, measures and their occurrence time are fully observed and (2) the partial measures of the future. Note that the situation cannot observe or control the occurrences of events beyond state k since we want to show that triggering r at state k is sufficient to cause a conflict regardless how the system responds after state k.

A situation is *feasible* with respect to a rule set *Rules* if the past up to state k (i.e., σ_0^k) does not *violate* any rule in *Rules*. To provide a formal definition of non-violation up to some state (k), we extend the semantics of SLEEC DSL to evaluate a trace up to a specified state k to check whether a *violation* of rule r has already occurred, denoted as $\sigma \vdash^k r$. We define the extended semantics, *bounded semantics*, in Fig. 4.
A rule r is situationally conflicting if there exists a feasible r-triggering situation that eventually causes a conflict.

Definition 2 (Situational Conflict)**.** Let a rule set *Rules* be given. A rule r in *Rules* is *situationally conflicting* if there exists an r-triggering *situation* $(\sigma_0^k, \vec{M}_k)$ such that: (1) $\sigma_0^k \vdash^k$ *Rules*, and (2) there *does not* exist an extension $\sigma = (\mathcal{E}_1, \mathbb{M}_1, \delta_1), \ldots (\mathcal{E}_n, \mathbb{M}_n, \delta_n)$ of σ_0^k, such that σ preserves the measures in $\vec{M}_k$, i.e., for every $i > k$, and every measure $m \in M$, either $\mathbb{M}_i(m) = \vec{M}_k[i-k](m)$ or $\vec{M}_k[i-k](m)$ is undefined), and σ fulfills the rules ($\sigma \in \mathcal{L}(Rules)$).

Remark 1. A rule r is vacuously conflicting in a rule set *Rules*, if and only if it is situationally conflicting for every r-triggering situation $(\sigma_0^k, \vec{M}_k)$.

3 TRANSLATION OF SLEEC DSL TO FOL*

In this section, we provide the translation function T in Tbl. 5, and prove the correctness claim of the translation prove the correctness of the translation in Thm. 1.

We now state and prove the correctness of the FOL* encoding:

Theorem 1 (Correctness of the FOL* translation). *Let a set of rules Rules and facts Facts in* SLEEC DSL *be given. There exists a trace $\sigma = (\mathcal{E}_1, \mathbb{M}_1, \delta_1), (\mathcal{E}_2, \mathbb{M}_2, \delta_2), \ldots (\mathcal{E}_n, \mathbb{M}_n, \delta_n)$ such that $\sigma \in$*

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

Figure 1: Function Norm takes *resp* and p and returns a set of normalized pseudo-rules.

Original SLEEC Rule

r_o = **when** e_1 **and** p_1 **then** e_2 **within** t_1 **otherwise** (e_3 **within** t_2 **unless** p_3 **then** e_4 **within** t_3)

Normalized SLEEC Rules

r_{n1} = **when** e_1 **then** ($p_1 \Rightarrow$ (e_2 **within** t_1)) **otherwise** (**not** $p_3 \Rightarrow e_3$ **within** t_2)

r_{n2} = **when** e_1 **then** ($p_1 \Rightarrow$ (e_2 **within** t_1)) **otherwise** ($p_3 \Rightarrow e_4$ **within** t_3)

Figure 2: 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}.

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

Figure 3: Semantics of normalized `SLEEC DSL` defined over trace $\sigma = (\mathcal{E}_1, \mathbb{M}_1, \delta_1) \ldots (\mathcal{E}_n, \mathbb{M}_n, \delta_n)$.

$\sigma \vdash_i^k ob$	iff	$\neg(\exists j \in [i,k].\sigma \not\models_i^k ob))$
$\sigma \vdash_i^k (p \Rightarrow ob)$	iff	$\sigma \models_i p \Rightarrow \sigma \vdash_i^k ob$
$\sigma \vdash_i^k cob^+$ **otherwise** $\bigvee_{cob}$	iff	$\sigma \vdash_i cob^+ \vee \exists j(j \leq k \wedge \sigma \not\models_i^j cob^+ \wedge \sigma \vdash_j^k \bigvee_{cob})$
$\sigma \vdash^k$ **when** e **and** p **then** $\bigvee_{cob}$	iff	$\forall i \in [1,k]((e \in \mathcal{E}_i \wedge \mathbb{M}_i(p)) \Rightarrow \sigma \vdash_i^k \bigvee_{cob})$

Figure 4: Bounded Semantics of normalized `SLEEC DSL` up to a time point k defined over trace $\sigma = (\mathcal{E}_1, \mathbb{M}_1, \delta_1) \ldots (\mathcal{E}_n, \mathbb{M}_n, \delta_n)$ where $n \geq k$.

$\mathcal{L}(Rules) \cap \mathcal{L}(Facts)$ if and only if $T(Rules) \wedge T(Facts) \wedge axiom_{mc}$ has a satisfying solution (D, v).

Sketch of Proof 1. *We prove the forward direction by constructing a satisfying solution (D, v) to $T(Rules) \wedge T(Facts)$ from the trace $\sigma \in \mathcal{L}(Rules) \cap \mathcal{L}(Facts)$. For every state $(\mathcal{E}_i, \mathbb{M}_i, \delta_i) \in \sigma$, we follow the construction rules: (1) for every event $e \in \mathcal{E}_i$, add a relational object o^e of class C^e such that $v(o^e.ext) = \top$ and $v(o^e.time) = \delta_i$; and (2) add a relational object o^M such that $o^M.time = \delta_i$ and $v(o^M.m) = \mathbb{M}_i(m)$ for every measure $m \in M$. We then prove that the constructed (D, v) is a solution to $T(Rules) \wedge T(Facts) \wedge axiom_{mc}$.*

We prove the backward direction by constructing σ from a satisfying solution (D, v) to $T(Rules) \wedge T(Facts) \wedge axiom_{mc}$. The construction maps every relational object o^e and o^M to some state $(\mathcal{E}_i, \mathbb{M}_i, \delta_i) \in \sigma$, where (1) $e \in \mathcal{E}_i$ if $v(o^e).ext = \top \wedge v(o^e).time = \delta_i$; and (2) $\mathbb{M}_m = v(o^M.m)$ for every $m \in M$ if $v(o^M).ext = \top \wedge v(o^M).time = \delta_i$. We then prove $\sigma \in \mathcal{L}(Rules) \cap \mathcal{L}(Facts)$ and conclude the proof.

4 FOL* ENCODING FOR SITUATIONAL CONFLICTS

In this section, we first define the states of obligation in Def. 3, and then use it to prove sufficient condition for situational conflict in Thm. 1. Next, we present the FOL* encoding for the sufficient condition of situational conflict in Tab. 6, and finally provide a sketch of the proof the correctness of the encoding (Thm. 2).

Definition 3 (State of Obligations)**.** Let a rule set *Rules*, a rule $r \in Rules$ and an r-triggering situation $(\sigma_0^k, \vec{M}_k)$ be given. The time point k is the last time point of σ_0^k, and it is also when r is triggered. The status of rules, obligation chains, conditional obligations and obligations are defined as follows:

Triggered: A rule $r =$ **when** $e \wedge p$ **then** $\bigvee_{cob}$ is *triggered* at time point i if $i \leq k$ and $\sigma_0^k \models_i e$ and $\sigma_o^k \models p$. If a rule $r =$ **when** $e \wedge p$ **then** $\bigvee_{cob}$ is triggered at i, then the obligation chain $\bigvee_{cob}$ is

$$
\begin{array}{ll}
T(\textbf{when } e \textbf{ and } p \textbf{ then } \bigvee_{cob}) & \rightarrow \forall o^e : C^e \exists o^M : C^M(o^m.time = o^e.time \wedge (T^*(p, o^M) \Rightarrow T^*(\bigvee_{cob}, o^M))) \\
T(\textbf{exists } e \textbf{ and } p \textbf{ while } \bigvee_{cob}) & \rightarrow \exists o^e : C^e \exists o^M : C^M(o^m.time = o^e.time \wedge (T^*(p, o^M) \wedge T^*(\bigvee_{cob}, o^M))) \\
T^*(cob_1 \textbf{ otherwise } cob_2 \ldots cob_n,\ o^M) & \rightarrow T^*(cob_1, o^M) \vee \exists o_j^M : C^M(\textbf{Violation}(cob_1, o^M, o_j^M) \wedge T^*(cob_2 \textbf{ otherwise } \ldots cob_n, o_j^M)) \\
T^*(\textbf{not } \bigvee_{cob},\ o^M) & \rightarrow \neg T^*(\bigvee_{cob}) \\
T^*(p \Rightarrow ob, o^M) & \rightarrow \neg T^*(p, o^M) \wedge T^*(ob, o^M) \\
\textbf{Violation}(p \Rightarrow ob, o^M, o_j^M) & \rightarrow T^*(p, o^M) \wedge \textbf{Violation}(ob, o^M, o_j^M) \\
T^*(e \textbf{ within } t, o^M) & \rightarrow \exists o^e : C^e(o^M.time \leq o^e.time \leq o^M.time + T^*(t, o^M)) \\
\textbf{violation}(e \textbf{ within } t, o^M, o_j^M) & \rightarrow o_j^M.time = o^M.time + T^*(t, o^M) \wedge \neg T^*(\textbf{not } e \textbf{ within } t, o^M) \\
T^*(\textbf{not } e \textbf{ within } t, o^M) & \rightarrow \neg T^*(\textbf{not } e \textbf{ within } t, o^M) \\
\textbf{Violation}(\neg e \textbf{ within } t, o^M, o_j^M) & \rightarrow \begin{array}{l} (o^M.time \leq o_j^M.time \leq o^M.time + T^*(t, o^M)) \wedge \exists o^e : C^e(o_j^M.time = o^e.time) \\ \neg(\exists o_1^e : C^e(o^M.time \leq o_1^e.time < o_j^M.time)) \end{array} \\
T^*(c, o^M) \rightarrow c & T^*(m, o^M) \rightarrow o^M.m \\
T^*(-t, o^M) \rightarrow -T^*(t, o^M) & T^*(t_1 + t_2, o^M) \rightarrow T^*(t_1, o^M) + T^*(t_2, o^M) \qquad T^*(c \times t, o^M) \rightarrow T^*(c, o^M) \times T^*(t, o^M) \\
T^*(\top, o^M) \rightarrow \top & T^*(t_1 > t_2, o^M) \rightarrow T^*(t_1, o^M) > T^*(t_2, o^M) \qquad T^*(t_1 = t_2, o^M) \rightarrow T^*(t_1, o^M) = T^*(t_2, o^M) \\
T^*(\neg p, o^M) \rightarrow \neg T^*(p, o^M) & T^*(p_1 \wedge p_2, o^M) \rightarrow T^*(p_1, o^M) \wedge T^*(p_2, o^M) \qquad T^*(p_1 \vee p_2, o^M) \rightarrow T^*(p_1, o^M) \vee T^*(p_2, o^M)
\end{array}
$$

Figure 5: Translation rules from SLEEC DSL to FOL*. Given a SLEEC DSL rule r, T translates r to an FOL* formula using the translation function T^*. The function T^* recursively visits the elements (i.e., term, proposition and obligations) of r and translates them into FOL* constraints under a relational object o^M representing the measures when r is triggered.

triggered at i. If an obligation chain $\bigvee_{cob} = cob$ **otherwise** $\bigvee'_{cob}$ is triggered at i, then (1) the *conditional obligation* cob is triggered at i and (2) $\bigvee'_{cob}$ is *triggered* at $j > i$ if cob is *violated* at j or cob is *blocked* at i. If a conditional obligation $p \Rightarrow ob$ is triggered at i and p is evaluated to $\top$ at i, then ob is triggered at i.

Fulfilled: An obligation ob is *fulfilled* at time point $j \leq k$ if it is triggered at some time point $i \leq j$ and $\sigma_0^j \models_i ob$. A conditional obligation $p \Rightarrow ob$ (triggered at i) is *fulfilled* at j if its obligation ob is fulfilled at j or p evaluated to $\bot$ at i. An obligation chain $\bigvee_{cob} = cob$ **otherwise** $\bigvee'_{cob}$ is fulfilled at j if cob is fulfilled at j or $\bigvee'_{cob}$ is fulfilled at j.

Violated: An obligation ob (*triggered* at i) is *violated* at a time point $j \leq k$ if $\sigma_0^k \not\models_i^j ob$. A conditional obligation $p \Rightarrow ob$ is *violated* at time point j if ob is *violated* at j. An obligation chain $\bigvee_{cob} = \bigvee'_{cob}$ **otherwise** cob is *violated* at time point j if $\bigvee'_{cob}$ is *violated* at some point $j' \leq j$, cob is *triggered* at j and cob is *violated* at j'.

Active: An obligation, conditional obligation and obligation chain are *active* at time point j if they are *triggered* at some time point $i \leq j$ and are not *fulfilled* and *violated* at any time point $j' \in [i, j]$.

Forced: An obligation chain $\bigvee_{cob}$ is *forced* at time point $j \geq k$ if $\bigvee_{cob}$ is *active* at j. If an obligation chain $\bigvee_{cob} = cob^+$ **otherwise** $\bigvee'_{cob}$ is *forced*, and $\bigvee'_{cob}$ is *blocked* at the time point j' when cob^+ expires (s.t., $\delta_{j'} = \delta_j + \mathbb{M}_j$), then cob^+ is *forced* at time point j. If a conditional obligation $p \Rightarrow ob$ is *forced* at j, and ob is triggered at j, then ob is *forced* at j.

Blocked: An obligation ob (triggered at i) is *blocked* at time point j if it is *active* and there is an obligation ob' such that (1) ob' is *forced* at time point j; (2) if $ob = e$ **within** t and $ob' = \neg e$ **within** t' then $\mathbb{M}_i(t) + \delta_i \leq \mathbb{M}_{i'}(t') + \delta_{i'}$; and (3) if $ob = \neg e$ **within** t and $ob' = e$ **within** t' then $\mathbb{M}_i(t) + \delta_i \geq \mathbb{M}_{i'}(t') + \delta_{i'}$. A conditional obligation $p \Rightarrow ob$ is *blocked* at j if ob is *blocked* j and p evaluates to $\top$ at j. An obligation chain cob^+ **otherwise** $\bigvee_{cob}$ is *blocked* at time point j if cob^+ is *blocked* at some time point j and $\bigvee_{cob}$ is *blocked* at time point j' when cob expires (s.t., $\delta_{j'} = \delta_j + \mathbb{M}_j$).

Noticed that there are circular dependencies between forced and blocked obligation chains. Fortunately, by encoding the status definitions into FOL*, we can leverage FOL* solver's ability to incrementally and lazily unroll the necessary definitions to resolve the dependencies.

The status of obligations at time point k is determined by the historical states leading up to k (i.e. the prefix of a trace σ_0^k) and the future measures after k (i.e., $\vec{M}_k$). When a rule is triggered at time point k while its response (the entire obligation chain) is blocked given a situation $(\sigma_0^k, \vec{M}_k)$, then a conflict is inevitable.

Lemma 1. *For every rule $r =$ **when** $e \wedge p$ **then** $\bigvee_{cob}$ in a rule set Rules, if there exists a r-triggering situation $(\sigma_0^k, \vec{M}_k)$ (see Def. 1) where the obligation chain $\bigvee_{cob}$ is blocked at time point k, then r is situationally conflicting with respect to the situation $(\sigma_0^k, \vec{M}_k)$.*

The Proof of Lemma 1.

Sketch of Proof 2. *Proof by contradiction, we assume there exists a trace σ such that σ is an extension to σ_0^k and is also consistent with $\vec{M}_k$. Since $\sigma \in \mathcal{L}(Rules)$, then σ fulfill the obligation chain $\bigvee_{cob} = (cob_1, \ldots cob_n)$ triggered at k ($\sigma \models_k \bigvee_{cob}$). Therefore, by the semantics of obligation chain fulfillment, either $\sigma \models_k cob_1$ or cob_1 is positive and there exists a time point $k' \geq k$ such that $\sigma \not\models_k^{k'} cob_1$ and $\sigma \models_{k'} (cob_2, \ldots cob_n)$*

Since $\bigvee_{cob}$ is blocked at k, by Def. 3, the obligation ob_1 in cob_1 is blocked at k, and for every conditional obligation cob_m, the obligation ob_m is cob_m is blocked at the unique time when ob_{m-1} is violated (the violation time is unique since $cob_1 \ldots cob_{n-1}$ are all positive). Therefore, it is sufficient to show that if an obligation ob is blocked, then σ does not fulfill ob_m. There are two cases:

*First, we consider the case $ob = e$ **within** t. There is an event occurred at some time point $k \geq j$ where $\delta_j \leq \delta_k \leq \delta_j + \mathbb{M}_j$. Since ob is blocked, then there exists an obligation forced by some rule r' (triggered at j') such that $ob' = \neg e$ **within** t' where $\mathbb{M}_{j'}(t') \geq \mathbb{M}_j$. Therefore, the occurrence of e at time point k violates r'. Contradiction.*

*The case where $ob = \neg e$ **within** t can be proved analogously.* □

Remark 2. The sufficient condition for situational conflict defined in Lemma 1 is not a necessary condition, as some situational conflicts do not require a rule's response to be blocked at the last state of the situation. Let's consider the set of rules $\{r_1, r_2, r_3, r_4\}$, where r_1 = **when** e_1 **then** e_2 **within** 5, r_2 = **when** e_3 **then** $\neg e_2$ **within** 4, r_3 = **when** e_4 **then** e_3 **within** 3, and r_4 = **when** e_1 **then** $\neg e_3$ **within** 1. The rule r_1 is situational conflicting in the situation $(\sigma^1, ^*)$ where $\sigma^1 = (r_1, r_2, r_3, r_4, \mathbb{M}_1, 1)$ because, according to r_1 and r_2, e_2 must occur within the interval $(4, 5]$. Additionally, based on r_3 and r_4, the event e_3 must occur at a time $t \in (1, 3]$. For all possible values of t, according to r_2, e_2 cannot occur within the interval $(t, t+4]$, which covers the interval $(4, 5]$ and thus conflicts with r_1. However, in the situation σ^1, the obligation of r_1 is not blocked at time point 1. We refer to the situational conflicts caused by forced obligations "after" the situation as"transitive situation conflicts". Identifying situations that lead to transitive situation conflicts is not easily expressed as satisfiability (e.g., we need to cover the entire range of t in the example) and is left as future work.

We present the FOL* encoding in Fig.6 to describe the situation for a rule to be situational conflicting. Given a set of rules *Rules* and a rule $r \in$ *Rules*, every satisfying solution to the FOL* formula TSC(*Rules*, r) represents a situation where r is situational conflicting. The top-level encoding TSC(r, *Rules*) is presented in part (1) of Tab.6, which describes the existence of a situation $(\sigma_0^k, \vec{M}_k)$ where r is triggered at the last state of σ_0^k (o^M). The situation should be non-violating ($\mathrm{T}_{\downarrow}(Rules, o^M.time)$) and should block the response of r ($\textsc{Blocked}(\bigvee_{cob}, o^M, o^M)$). The FOL* encoding for non-violating and obligation blocking is presented in parts (2) and (3) of Tab. 6, respectively.

Note that eagerly expanding the definition of obligation blocking blows up the size of encoding exponentially (with respect to the number of obligations in *Rules*) due to the transitive dependencies between blocked obligations and forced obligations. To avoid the blow-up, we lazily expand the definition of obligation blocking by introducing an *internal* class of relational object $C^{blockob}$ for every obligation *ob* in *Rules* to indicate if and when *ob* is blocked. The axiom *axiomBlockob* is added to describe the definition of a blocked obligation (Def. 3) and is lazily applied to relational objects of $C^{block_{ob}}$ in a given domain.

Theorem 2 (Encoding of Situational Conflict). *Let a rule set Rules be given. For every rule $r \in$ Rules, if the FOL* formula TSC(r, Rules) is satisfiable, then r is situationally conflicting.*

Sketch of Proof 3. *If (D, v) is a satisfying solution to TSC(r, Rules), then we use the same method in the proof of Thm. 1 to construct a situation σ from (D, v). We then show that the encoding in Tab. 6 conforms with the semantics of non-violation and the status of obligations (in Def. 3). Finally, we show that the constructed σ satisfies the sufficient condition for situational conflict (Lemma. 1), and thus r is situational conflicting w.r.t σ.* □

5 FOL* * PROOF OF UNSAT

A causal FOL* proof is a sequence of derivation steps $L_1, L_2 \ldots, L_n$. Each step L_i is a tuple (i, ψ, o, *Deps*) where (1) i is the ID of the derivation step; (2) ψ is the derived FOL* lemma; (3) o is the name of the *derivation rule* used to derive ψ; and (4) *Deps* are IDs of dependent lemmas for deriving ψ. A *derivation step is sound* if the lemma ψ can be obtained via the derivation rule o using lemmas from *Deps*. A *proof is sound* if every derivation step is sound. The proof is *refutational* if the final derivation contains the lemma $\bot$ (UNSAT).

Table 1: FOL* causal proof of UNSAT for ϕ_1 and ϕ_2 in Ex. 2.

ID	Lemma	Derivation Rule	Deps
1	$\forall a : A \exists b : B \cdot (a.time \le b.time \le a.time + 10)$	**INPUT**: ϕ_1	{}
2	$\exists a : A \forall b : B \cdot (b.time \ge a.time + 20 \wedge p(a, b))$	**INPUT**: ϕ_2	{}
3	$\forall b : B \cdot (b.time \ge a_1.time + 20 \wedge p(a_1, b))$	**EI**: $[a \leftarrow a_1]$	{2 }
4	$\exists b : B \cdot (a_1.time \le b.time \le a_1.time + 10)$	**UI**: $[a \leftarrow a_1]$	{1, 3}
5	$a_1.time \le b_1.time \le a_1.time + 10$	**EI**: $[b \leftarrow b_1]$	{4}
6	$b_1.time \ge a_1.time + 20 \wedge p(a_1, b_1)$	**UI**:$[b \leftarrow b_1]$	{3, 5}
7	$b_1.time \ge a_1.time + 20$	**And**	{6}
8	$\bot$	**Impl**	{5, 7}

Example 2. Let FOL* formulas, $\phi_1 : \forall a : A \exists b : B \cdot (a.time \le b.time \le a.time + 10)$ and $\phi_2 : \exists a : A \forall b : B \cdot (b.time \ge a.time + 20 \wedge p(a, b))$ be given, where A and B are classes of relational objects, *time* is an attribute of type $\mathbb{N}$ and p is a complex predicate. $\phi_1 \wedge \phi_2$ is UNSAT, and the proof of UNSAT is shown in Tbl. 1. The proof starts by introducing the **Input** ϕ_1 and ϕ_2 (steps 1, 2), and then uses existential instantiation (**EI**) in steps 3, 5 and universal instantiation (**UI**) in steps 4, 6 to eliminate quantifiers in ϕ_1 and ϕ_2. In step 7, the **And** rule decomposes the conjunction and derives part of it as a new lemma. Finally, in step 8, $\bot$ is derived with the **Impl** rule from the quantifier-free (QF) lemmas derived in steps 5 and 7. The proof is refutational because step 8 derives $\bot$.

5.1 Derivation Tree and proof reduction

Given a refutation proof, one can construct a *derivation graph* where every lemma is a node and its dependencies are the incoming edges to the node. The roots of the derivation graph are the input formulas and axioms (e.g., $axiom_{mc}$) and the (only) leaf of the graph is the derived $\bot$.

Using the derivation graph, one can check the soundness of the proof and reduce it by traversing the graph backwards from the leaf (step 8 in Fig. 1). While visiting a node, we first check if the lemma represented by the node can be soundly derived using the derivation rule with the lemmas in its dependency, and then reduce the dependency if not every lemma is necessary. Only the nodes representing the lemmas in the reduced dependencies are scheduled to be visited in the future. For instance, after checking the derivation step 7, its dependencies, 5 and 8, are scheduled to be visited next. If every scheduled node is visited without any failure, the proof is successfully verified, and the visited portion of the graph constitutes *the reduced proof*. In a reduced proof, every derived lemma is used to derive $\bot$.

We use two special derivation rules, **Input** and **Implication**. The rule **Input** adds an input FOL* formula as a fact to the proof. The rule **Implication** derives new QF lemmas via logical implication, and it can be verified using an SMT solver by solving the formula $deps \wedge \neg C$ where C is the derived lemmas and *deps* are lemmas in the implication step's dependencies. The derivation is *valid* if and only if the formula is UNSAT, and the UNSAT core returned by the SMT solver becomes the reduced dependencies.

Example 3 (Reduced Dependencies)**.** Let $L1 : A > B$, $L2 : B > C$ and $L3 : C > 5$ be three (derived) lemmas. Suppose a lemma $L4$:

TSC(**when** $e \wedge p$ **then** $\bigvee_{cob}$, *Rules*)	$\rightarrow \exists o^e : C^e \exists o^M : C^M (o^M.time = o^e.time \wedge T^*(p, o^M) \wedge$ Blocked$(\bigvee_{cob}, o^M, o^M)$ $\wedge$T$_{\downarrow}$(*Rules*, $o^M.time$)) $\wedge$ *axiom*$_{mc}$ $\wedge$ *axiomBlock*$_{ob}$ for every obligations *ob* in *Rules*
T$_{\downarrow}$(**when** $e \wedge p$ **then** $\bigvee_{cob}$, *end_time*)	$\rightarrow \forall o^e : C^e (o^e.time \leq end\_time \Rightarrow \exists o^M : C^M$ $(o^m.time = o^e.time \wedge (T^*(p, o^M) \Rightarrow$ T$^*_{\downarrow}(\bigvee_{cob}, o^M, end_time))))$
T$^*_{\downarrow}$(*ob*$_1$ **otherwise** ... *ob*$_n$, o^M, *end_time*)	$\rightarrow$ T$^*_{\downarrow}(ob_1, o^M, end_time) \vee \exists o_v^M : C^M($**violation**$(ob_1, o^M, o_v^M) \vee$ T$^*_{\downarrow}(ob_n, o_v^M, end_time))$
T$^*_{\downarrow}$(e **within** t, o^M, *end_time*)	$\rightarrow \exists o^e : C^e (o^M.time \leq o^e.time \leq o^M.time + T^*(t, o^M) \ \vee \ o^M.time + T^*(t, o^M) > end\_time$
T$^*_{\downarrow}$($\neg e$ **within** t, o^M, *end_time*)	$\rightarrow \neg(\exists o^e : C^e (o^M.time \leq o^e.time \leq$ Min$(o^M.time + T^*(t, o^M),, end_time))$
Blocked$((p \Rightarrow e$ **within** $t)$ **otherwise** $\bigvee_{cob}, o_i^M, o_c^M)$	$\rightarrow$ Blocked$((p \Rightarrow e$ **within** $t), o_i^M, o_c^M) \wedge \exists o_v^M : C^M$ $(o_v^M.time = o_i^M.time + T * (t, o_i^M)) \wedge$ Blocked$(\bigvee_{cob}, o_v^M, o_c^M)$
Blocked$(p \Rightarrow ob, o_i^M, o_c^M)$	$\rightarrow T^*(p, o_i^M) \wedge$ Blocked(ob, o_i^M, o_c^M)
Blocked(ob, o_i^M, o_c^M)	$\rightarrow \exists o^{block_{ob}} : C^{block_{ob}} (o^{block_{ob}}.i = o_i^M.time \wedge o^{block_{ob}}.c = o_c^M.time)$
axiomBlock$_{ob}$	$\rightarrow \forall o^{block_{ob}} : C^{block_{ob}} \exists o_i^M, o_c^M : C^M$ $(o_i^M.time = o^{block_{ob}}.i \wedge o_c^M.time = o^{block_{ob}}.c \wedge$ _Blocked$(ob, o_i^M, o^{M_c}))$
_Blocked$(e$ **within** $t, o_i^M, o^{M_c})$	$\rightarrow$ Active$(e$ **within** $t, o_i^M, o^{M_c}) \wedge \bigvee_{ob \in OBG(\neg e)} (\exists o_1^M : C^M (o_1^M.time \leq o^M.time)$ $\wedge$Forced$(ob, o_1^M, M_c) \wedge (o^M.time + T^*(t, o^M) \geq o_1^M.time + T^*(t_1, o^M))$ where t_1 is *ob*'s time limit
_Blocked$(\neg e$ **within** $t, o_i^M, o^{M_c})$	$\rightarrow$ Active$(\neg e$ **within** $t, o_i^M, o^{M_c}) \wedge \bigvee_{ob \in OBG(e)} (\exists o_1^M : C^M (o_1^M.time \leq o^M.time)$ $\wedge$Forced$(ob, o_1^M, M_c) \wedge (o^M.time + T^*(t, o^M) \leq o_1^M.time + T^*(t_1, o^M))$ where t_1 is *ob*'s time limit
Active(ob, o_i^M, o_c^M)	$\rightarrow$ Triggered$(ob, o_i^M) \wedge \neg$Violated$(ob, o_i^M, o_c^M) \wedge \neg$Fulfilled$(ob, o_i^M, o_c^M)$
Fulfilled$(e$ **within** $t, o_i^M, o_c^M,,)$	$\rightarrow T^*(e$ **within** Min$(t, o_c^M.time)), o^M$
Fulfilled$(\neg e$ **within** $t, o_i^M, o_c^M)$	$\rightarrow$ T$^*_{\downarrow}(\neg e$ **within** $t, o^M, o_c^M.time)$
Violated(ob, o_i^M, o_c^M)	$\rightarrow$ Fulfilled(Noncomp$(ob), o_i^M, o_c^M.time,,)$
Triggered(ob, o^M)	$\rightarrow$ let trigger_rule$(ob) =$ **when** $e \wedge p$ **then** $\bigvee_{cob}$ where $(p_m \Rightarrow ob) = \bigvee_{cob}[m]$ if $m = 1$ then $\exists o^e : C^e (o^e.time = o^M.time \wedge T^*(p \wedge p_m, o^M))$ else $\exists o_i^M : C^M($Violated$(\bigvee_{cob}[m-1], o_i^M, o^M) \vee$ Blocked$(\bigvee_{cob}[m-1], o_i^M, o^M)))$
Forced$(p \Rightarrow ob, o_i^M, o_c^M)$ where $cob = \bigvee_{cob}[m]$	$\rightarrow T^*(p, o_i^M) \wedge$ Active$(ob, o_i^M, o_c^M) \wedge \neg($Blocked$(ob, o_i^M, o_c^M)) \wedge$ $\exists o_v^M : C^M($**Violation**$(ob, o_i^M, o_v^M) \wedge$ Blocked$(\bigvee_{cob}[m+1:], o_v^M, o_c^M))$
$OBG(h)$	$= \{ob \mid (h$ **within** $t)$ in the rule set$\}$
trigger_rule$(ob) =$ **when** e **then** $\bigvee_{cob}$	if and only if $ob \in \bigvee_{cob}$

Figure 6: The FOL* encoding TSC$(r, Rules)$ that describes a situation where the rule $r \in Rules$ to be situational conflicting. The table contains three parts: (1) the top-level encoding that describes the existence of a situation where r is triggered at the last state (o^M); (2) the FOL* constraint for describing the situation is non-violating (i.e., T$_{\downarrow}$($Rules, o^M.time$)); and (3) the FOL* encoding for describing blocked obligations (i.e., Blocked$(\bigvee_{cob}, o_i^M, o_c^M)$) as well as another status of obligations where $o^M{}_i$ is the state when the obligation is triggered and $o^M{}_c$ is the last state of the situation.

$A > C$ is derived using the **implication** rule given the dependencies $L1$, $L2$, $L3$. The rule can be verified by checking the satisfiability of $L1 \wedge L2 \wedge L3 \wedge \neg L4$. The result is UNSAT with an UNSAT core $L1$, $L2$ and $\neg L4$. Therefore, the reduced dependencies are $L1$ and $L2$.

5.2 Condition for involved atomic elements

In addition to reporting a binary "yes" or "no" answer to the satisfiability, the FOL* satisfiability checker LEGOS also provides a causal proof of UNSAT if the encoded formula is unsatisfiable. We project the proof into the input SLEEC DSL rules to highlight the causes of WFI problems at the level of *atomic elements*. More specifically, we want to highlight every *involved atomic elements* in the proof.

Definition 4 (Atomic element)**.** An *atomic element* in SLEEC DSL is one the followings: (1) an *atomic proposition ap*: $\top \mid \bot \mid t = t \mid t \geq t \mid \neg ap$, (2) a triggering event e (where e in **when** $e \wedge \ldots$ **then** ... or **exists** $e \wedge \ldots$ **while** ...), (3) a response event e' in an obligation (in e' **within** ...), or (4) a deadline t of an obligation (in ... **within** t).

Definition 5 (Involved atomic element)**.** Let a proof L be given. We denote $Imp(L)$ as the set of QF lemmas derived via or listed as dependencies for the derivation rule **Impl**. An atomic proposition ap is *involved* if $Imp(L)$ contains the quantifier-free (QF) formula $T^*(ap, o^M)$ for some relational object of class C^M where T^* is the translation function for SLEEC DSL element defined in Tab. 5. A triggering event e for "**when** $e \wedge p$ **then** $\bigvee_{cob}$" is involved if $Imp(L)$ contains a QF formula $\neg(o^e.ext) \vee \neg T^*(p, o^M) \vee F$ where F is a QF formula, and both o^e and o^M are relational objects of class C^e and C^M, respectively. Similarly, a triggering event e for "**exists** $e \wedge p$ **while** $\bigvee_{cob}$" is involved if $Imp(L)$ contains a formula $o^e.ext \wedge T^*(p, o^M) \wedge F$. An obligation head e for e **within** t is involved if $Imp(L)$ contains a QF lemma $o^e.ext \wedge o^e.time \geq o^M.time \wedge F$ for some object o^e and o^M. An obligation deadline t for e **within** t is involved if $Imp(L)$ contains a quantifier-free lemma $o^e.time \leq o^M.time + T^*(t, o^M) \wedge F$ for some object o^e and o^M.

5.3 Evaluation Protocol

The objective of the evaluation protocol is twofold. First, it aims to assess the relevance of identified WFI issues, including vacuous-, situational-conflicts, redundancy, insufficiency, and restrictiveness, within each case study. Second, it aims to evaluate the usability of the diagnosis for resolving the identified WFI issues. For each type of issue identified, the following questions should be posed to the stakeholders:

Relevance of WFI issues and diagnostics.

(1) Do you understand the WFI issues? Please respond with 'yes', 'no', or 'not sure'.
(2) If 'yes', is the issue relevant or spurious? If you consider it spurious, please provide a justification.
(3) If the WFI issue is not considered spurious, was the diagnostic information useful for understanding the problem? Please respond with 'yes' or 'no'.
(4) If the WFI issue is not considered spurious, can the diagnosis be used to resolve it? If so, please provide details on how it can be resolved.

(5) Which aspect of the diagnostic was unhelpful? Was it the highlighted rules, the trace, or the measure values?
(6) If the inconsistency cannot be resolved, is it due to the diagnosis lacking sufficient assistance? If so, please specify what is lacking. Alternatively, is it determined to be impossible to resolve the inconsistency? If so, please explain the reason.
(7) Is there an WFI issue that you believe we did not capture?

Global feedback.

- Do you have any feedback regarding the strengths of the proposed approach?
- Do you have any feedback regarding the weaknesses of the proposed approach?
- Do you have any feedback regarding the strengths of the produced diagnosis?
- Do you have any feedback regarding the weaknesses of the produced diagnosis?
- Do you have any additional needs you encountered that could benefit from formal automated tool assistance?