Mapping Business Rules to LTL Formulas

Isaac Mackey and Jianwen Su

Dept. of Computer Science, University of California at Santa Barbara
isaac_mackey@ucsb.edu, su@cs.ucsb.edu

1 Introduction

A business service consists of a set of business processes. The set of process instances serving a client forms a service enactment. Business service rules are conditions restricting enactments to comply with policies and regulations and to honor service-level agreements with clients. The problem of service provisioning includes specifying business service rules, monitoring enactments to detect violations of rules at run-time, and maintaining and updating rules. We develop an approach towards service provisioning by (1) modeling rules in a logic language and (2) automatically generating finite state machines (FSMs) as run-time monitors from a given set of rules. This approach involves translating a subclass of rules called “simple” rules to linear temporal logic (LTL) formulas, where known algorithms [6] translate LTL formulas to FSMs. The central problem in our approach is the following rule translation problem: Given a rule over a service, construct an equivalent LTL formula.

This work is related to monitoring runtime behaviors of web services and business processes specified in temporal logics. For example, [2] turned Declare [4] into colored finite state automata for runtime verification. [3] extended the Declare framework with quantitative time constraints, mapping this extension into event calculus. Notably in our approach, a monitor has a fixed size depending on the size of the rule.

We model services and rules as follows. A service schema $S$ is a finite set of process names. Each process instance is tagged with a timestamp for the completion (or initiation) of the process instance. A service enactment $\eta$ of a service schema $S$ is a mapping $\eta : S \rightarrow 2^N$ such that for each $p \in S$, $\eta(p)$ is a finite set representing the timestamps of instances of $p$. Business rules are formula constructed by the following logic language. A (timed) process atom is an expression "$p@x$", where $p$ is a process name and $x$ is a variable, that indicates an instance of process $p$ happens at timestamp $x$. A gap atom is an expression "$x \leq_n y$" or "$x \geq_n y$" where $x, y$ are variables and $\leq_n, \geq_n (n \in \mathbb{Z})$ are predicates. A constraint is a finite conjunction of atoms. A rule is a statement of implication from one constraint to another. The intent of a rule $\phi \rightarrow \psi$ is to require that each set of process instances satisfying one constraint $\phi$ can be extended to satisfy another constraint $\psi$.

We translate rules into linear temporal logic with past operators, defined below. LTL formulas are built recursively:

$\varphi := p \mid \text{true} \mid \text{false} \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \rightarrow \varphi \mid X \varphi \mid Y \varphi \mid F \varphi \mid P \varphi$
where \( p \in S \), \textit{true}, \textit{false} are Boolean values, and \( \neg, \lor, \land, \rightarrow \) are Boolean operators, and \( X \) (\textit{next}), \( F \) (\textit{future}), \( Y \) (\textit{yesterday}), and \( P \) (\textit{past}) are temporal operators \([5],[1]\).

The following notions are used for convenience: \( X^i \) \((i \in \mathbb{Z})\) means \( i \) consecutive \( X \) operators when \( i \geq 0 \) and \( i \) consecutive \( Y \) operators when \( i < 0 \).

Given an enactment \( \eta \) of a service schema \( S \), the trace \( \pi_\eta \) of \( \eta \) is defined as follows: if \( \kappa \) be the largest timestamp in \( \eta \), then \( \pi_\eta = \pi_\eta[0]...\pi_\eta[\kappa] \) where for each \( i \in [0, \kappa] \) and each \( p \in S \), \( \pi_\eta[i](p) = \text{true} \) if \( i \in \eta(p) \). The technical problem studied in this paper can be stated: Given a set of rules \( R \) over a service schema \( S \), is there an LTL formula \( \varphi \) such that for each enactment \( \eta \) of \( S \), \( \eta \) satisfies every rule in \( R \) iff \( \pi_\eta, 0 \models \varphi \)?

2 Mapping Rules to LTL

Consider a business rule named Initial Deposit that states: each client should make a payment no later than three days after a \textit{Schedule} process responds to the client’s request. In our model, this rule is specified as:

\[
\begin{align*}
\text{\texttt{r}id}: & \{\text{Request}@x, \text{Schedule}@y, x \leq 0 \ y\} \rightarrow \{\text{Payment}@z, y \leq 0, y \geq 3 \ z\} \\
\end{align*}
\]

Let \( \varphi_l \) and \( \varphi_r \) be the constraint at the left- and right-hand-side (resp.) of \( \text{\texttt{r}id} \). There is a natural (faithful) representation of \( \varphi_l \) and \( \varphi_r \) as acyclic, undirected graphs, shown below in Fig. 1 as two graphs connected by a dashed line.

![Fig. 1: A tree of the Initial Deposit rule, where y is a shared variable](image)

We translate each into LTL with respect to \( y \), denoted as \( \tau_{\varphi_l,y} \), yielding

\[
\tau_{\varphi_l,y} = \text{Schedule} \land P \text{Request}, \quad \tau_{\varphi_r,y} = \text{Schedule} \land \bigvee_{0 \leq i \leq 3} X^i \text{Payment}
\]

\( \text{\texttt{r}id} \) expresses a property of all assignments that select appropriate timestamps of \text{Request} and \text{Schedule} instances. This corresponds to a property of all instants of a trace that satisfy \text{Schedule} \land P \text{Request}. To reflect this coverage, the implication is placed in the scope of the global operator \( G \). The translation of \( \text{\texttt{r}id} \) is

\[
\text{\texttt{r}id}^{\text{LTL}} : G((\text{Schedule} \land P \text{Request}) \rightarrow \bigvee_{0 \leq i \leq 3} X^i \text{Payment})
\]

A rule \( \phi \rightarrow \psi \) (over a service schema) is \textit{simple} if \( \phi \cup \psi \) is acyclic and there is exactly one variable shared by \( \text{var}(\phi) \) and \( \text{var}(\psi) \). Let \( r : \phi_l \rightarrow \phi_r \) be a simple rule with a shared variable \( y \). Then the translation \( \gamma(r) \) of \( r \) is \( G(\tau_{\phi_l,y} \rightarrow \tau_{\phi_r,y}) \).

Let \( S \) be an arbitrary service schema and \( R \) an arbitrary set of simple rules over \( S \). It can be shown that all enactments \( \eta \) of \( S \), \( \eta \) satisfies \( R \) iff \( \pi_\eta, 0 \models \bigwedge_{r \in R} \gamma(r) \).
References