Group 8: Modelling and Reasoning with Business Processes and Workflows

Nahid Mahbub, FBK, Trento, Italy
Robert Muthuri, Erasmus Mundus LAST-JD, Turin, Italy
Stefan Scheglmann, Univ. Koblenz, Germany
Ognjen Savković, FUB, Bozen-Bolzano, Italy
Laura Genga, Univ. Politecnica delle Marche, Ancona, Italy
What are Business Processes?

• **Business Processes** (BPs) are set of activities organized to accomplish a specific **goal**
  - E.g., Order-Delivery, Production chain, etc.

• Business Processes are **used** for
  - Documentation
  - Communication
  - Execution
  - Static Analysis
    - Verification of Properties
    - Simulation and performance analysis
    - Comparability check, etc.
BPs Languages

• ‘04 BPEL: Business Process Execution Language
  o **executable language** for specifying actions within business processes with **web services**

• ‘05 BPMN: Business Process Modeling Notation
  o **graphical** modeling language
  o **de facto standard**
Static Analysis of BPs

- ‘97 Verification of Workflow Nets, van der Aalst
  - Semantics via Petri Nets
  - Checking for deadlocks, reachability, etc.

- ‘03 Workflow Patterns, van der Aalst
  - Exhaustive analysis of control-flow, resource, and exception handling
Business Processes and Data

• How the **data impacts** on **process execution**? E.g., can I buy an item that is **not available** at the warehouse (database)?

• But Data and Process modelling are **usually separated**!
BPs and Data: History

Relational Transducers, Abiteboul and Vianu, 2000
‘00

ASM transducers, Spielmann
‘03

Specification and verification of data-driven Web applications, Deutsch et al.
‘07

Data-Centric Dynamic Systems, Calvanese et al., 12
‘12

Business Artifacts @ IBM/Nigam and Caswell
Business artifacts: An approach to operational specification

A. Nigam
N.S. Caswell

IBM System Journal 2003
Operational Specification (OPS)

- IFF (Information, Function, Flow)

- Targets
  - analyze
  - manage
  - control

- Business people
  - retains formality for
    - reasoning
    - automated implementation
Business Artifacts

“Business artifacts constitute concrete information chunks that the business creates and maintains”

• Two parts
  o enterprise-wide unique identity
  o self-describing content (Key, Value)

• Identity unchangeable, Unsplittable
  o multiple artifacts with same content but different id

• Manipulatable
  o update
  o copy from other artifacts
  o adds from any source (input, computation, ...)
guest-check(
  ID 123
  context ()
  customer (num 3)
  store (ID(55) server(2)
    item (desc Hamb price 2.57
      cooked "13:23 04/07/1998"
    )
    delivered "13:26 04/07/1998"
  tax 0.33
  tender ( total 2.9
    cash 20.00 coupon 1.00
    change 18.10)
)
Artifact Life Cycle

- each artifact has a lifecycle
- end-to-end processing
  - creation
  - completion
  - archiving
- Places
  - Tasks (changes to artifacts)
  - Repositories (artifacts await further processing)
- describes the operations of a business
  - Function: how to add/update information
  - Flow: transport across functional units
Functions/Tasks

• Performs actions

• Activated by incoming artifact or externally

• Transforms artifacts (one or more)

• Artifacts are received or requested from repo

• After tasks completes all artifacts are ejected
Flow connector

• A pipe

• Ensures reliable transport

• for repos provides request/response communication
Modeling with Artifacts

- Pick key artifacts, construct lifecycle
- Create a candidate list (all artifacts needed for key artifact)
- Repeat
  - Take artifact from candidate list
  - construct lifecycle
  - add newly emerging artifacts to candidate list
Specification and Verification of Data-driven Web Applications

Alin Deutsch, Liying Sui and Victor Vianu

Journal of Computer and System Sciences, 2007
Data-driven Web Application
Modeling Web Application

- Action := f (Page, DB, State, Inputs)
- State := State of the application (set of relations)
- Inputs := Interaction with outside world (users or apps)

- Output relations := Application response
  - e.g., which item is selected for purchase
A Web application $W$ is a tuple $\langle D, S, I, A, W, W_0, W_\epsilon \rangle$, where:

- $D, S, I, A$ are relational schemas:
  - Database schema
  - State schema
  - Input schema
  - Action schema
- $W$ is a finite set of Web page schemas
- $W_0 \in W$ is the home page schema, and
  $W_\epsilon \not\in W$ is the error page schema
A Web page schema $W \in \mathbf{W}$ is a tuple $\langle I_W, A_W, T_W, R_W \rangle$, where:

- $I_W \subseteq I$, input set of the web page schema
- $A_W \subseteq A$, action set of the web page schema
- $T_W \subseteq T$, the set of target web pages
- $R_W$, the set of rules containing:
  - Input rules for each input relations $I \in I_W$
  - State rules for each state relations $S \in S$
  - Action rules for each action relations $A \in A_W$
  - Input rules for each input relations $V \in T_W$
Input and State Rules

- **Input rules:** $Options_I(\tilde{x}) \leftarrow \varphi_{I,W}(\tilde{x})$ where $Options_I$ is a relation of arity $k$, $\tilde{x}$ a $k$-tuple of distinct variables and $\varphi_{I,W}(\tilde{x})$ an FO formular over $D \cup S \cup \text{Prev}_I \cup \text{const}(I)$ with free $\tilde{x}$

- **State rules:** one, both or none of the following:
  - insertion rule $S(\tilde{x}) \leftarrow \varphi^+_S,W(\tilde{x})$
  - deletion rule $\neg S(\tilde{x}) \leftarrow \varphi^-_{S,W}(\tilde{x})$, with $S$ is of arity $k$, $\tilde{x}$ a $k$-tuple of distinct variables and $\varphi_{S,W}(\tilde{x})$, $\epsilon \in \{+, -, \}$ are FO formulars over schema $D \cup S \cup \text{Prev}_I \cup \text{const}(I) \cup I_W$, with free $\tilde{x}$
Action and Target Rules

- **Action rules:** $A(\tilde{x}) \leftarrow \varphi_{A,W}(\tilde{x})$, where $A$ is of arity $k$, $\tilde{x}$ a $k$-tuple of distinct variables and $\varphi_{A,W}(\tilde{x})$ an FO formular over schema $D \cup S \cup \text{Prev}_I \cup \text{const}(I) \cup I_W$ with free variables $\tilde{x}$

- **Target rules** $V \leftarrow \varphi_{V,W}$ where $\varphi_{V,W}$ is an FO sentence over schema $D \cup S \cup \text{Prev}_I \cup \text{const}(I) \cup I_W$

$W_ε = \langle \emptyset, \emptyset\{W_ε\}, R_W \rangle$ where $R_{W_ε}$ consists of the rule $W_ε \leftarrow \text{true}$. 
A run of a Web app $W$ over a database $D$ is an infinite sequence of configurations $\{\langle V_i, S_i, I_i, P_i, A_i \rangle \}_{i \geq 0}$ such that configurations $\langle V_i, S_i, I_i, P_i, A_i \rangle$ and $\langle V_{i+1}, S_{i+1}, I_{i+1}, P_{i+1}, A_{i+1} \rangle$ satisfy all Web app state, input, action and target rules accordingly.
Running Example

- Imagine a e-commerce Web site selling PCs (like Amazon.com)

- Allowed actions can be
  - New customer register with username and pass
  - Returning customers can login
  - Customer can search for PCs
  - Add found item to the shopping cart
  - Pay items from the shopping cart, etc.
Example: Pages

Pages in the running example

• **HP** - the home page
• **RP** - the new user registration page
• **CP** - the customer page
• **AP** - the administration page
• **LSP** - the laptop search page
• **PIP** - the product item page, products returned by search
• **CC** - the cart content
• **MP** - the error message page
Example: Home Page

Page $HP$

Inputs $I_{HP}: name, password, button(x)$

InputRules:

$\text{Options}_{button}(x) \leftarrow x = \text{"login"} \lor x = \text{"register"} \lor x = \text{"clear"}$

StateRules:

$error(x) \leftarrow \neg user(name, password) \land button(\text{"login"})$

$\land x = \text{"failed login"}$

TargetWebPages $T_{HP}: HP, RP, CP, AP, MP$

TargetRules:

$HP \leftarrow button(\text{"clear"})$

$RP \leftarrow button(\text{"register"})$

$CP \leftarrow user(name, password) \land button(\text{"login"}) \land name \neq (\text{"Admin"})$

$AP \leftarrow user(name, password) \land button(\text{"login"}) \land name = (\text{"Admin"})$

$MP \leftarrow \neg user(name, password) \land button(\text{"login"})$

EndPage $HP$
Example: Laptop Search Page

Page \( LSP \)

Inputs \( I_{LSP} : \text{laptopsearchpage}(ram, \text{hd}, \text{display}), \text{button}(x) \)

InputRules:

\[
\text{Options}_{\text{button}}(x) \leftarrow x = \text{“search”} \lor x = \text{“viewcart”} \lor \\
x = \text{“logout”}
\]

\[
\text{Options}_{\text{laptopsearch}}(r, h, d) \leftarrow \text{cirteria(“laptop”, “ram”, r)} \\
\quad \land \text{cirteria(“laptop”, “hdd”, h)} \land \text{cirteria(“laptop”, “display”, d)}
\]

StateRules:

\[
\text{userchoise}(r, h, d) \leftarrow \text{laptopsearch}(r, h, d) \land \text{button(“search”)}
\]

TargetWebPages \( T_{HP} : HP, PIP, CC \)

TargetRules:

\[
HP \leftarrow \text{button(“logout”)}
\]

\[
PIP \leftarrow \exists r \exists h \exists d \text{laptopsearch}(r, h, d) \land \text{button(“search”)}
\]

\[
CC \leftarrow \text{button(“view cart”)}
\]

EndPage \( LSP \)
Verification Language

• Verification of temporal aspects of web application
  o Verify properties over all runs of the web app
• Ex 1. “If page Ordered is reached in the run then page Payment is reached eventually”
• Ex 2. “Any shipped product is previously paid”

Since we have relations we need FO Temporal Logics
Linear Temporal Properties

- LTL-FO for checking linear properties
  - i.e., satisfied by all runs of a Web app

- E.g., “Any shipped product is previously paid”

\[
\forall pid, price \ [\xi(pid, price)] \ \Box \neg (\text{conf}(\text{name, price}) \land \text{ship}(\text{name, pid}))
\]

where \(\xi(pid, price)\) is the formula

\[
PP \ \land \text{pay}(price) \land \text{button(“authorized payment”)} \\
\land \text{pick}(pid, price) \\
\land \exists \text{pname catalog}(pid, price, pname)
\]
Branching Temporal Prop.

• CTL-FO (CTL*-FO) for checking branching time properties

• E.g., “a bought product will be eventually shipped, but until then, the user can still cancel the order”

\[ \forall pid \forall price \text{AG}(\xi(pid, price) \rightarrow A((\text{EF} cancel(name, pid) \cup \text{ship(name, pid)}))) \]
where \( \xi(pid, price) \) is the formula

\[ PP \land \text{pay}(price) \land \text{button}(“authorized \ payment”) \land \text{pick}(pid, price) \land \text{prod_price}(pid, price) \]
Undecidability

- Given Web App W and Temporal FO $\varphi$, we want to check whether $W \models \varphi$

- Undecidability follows immediately :)  
  - $\varphi$ is an FO sentence over D  
  - action rule $A \leftarrow \varphi$, where $A$ is a proposition  
  - $\varphi$ is finitely satisfiable iff $A \models \neg G \neg A$  
  - Trakhtenbrot’s theorem: finite satisfiability of FO sentences is undecidable
Gaining Decidability

- To gain decidability restrict FO formulas to
  - “input-bounded” quantification (restricted FO)
  - all state atoms are ground

- N.B. Web App model is nothing else but a compact representation of a transition system (or any other BPs and Data model)

- Model Checking?
Verif. via Model Checking

- Model checking technology requires the transition system to be finite
- However, here states are modeled relationally (not propositionally)

⇒ Infinite State Transition System
Verif. via Model Checking

(2)

- Restrict FO formulas to be "Input-bounded"
  - "Input-bounded" restricts quantification and helps to establish finitely many "isomorphic" configurations for a given LTL formula

- Then we can use "classical" model checking techniques

-CTL (CTL*) needs more restrictions to gain decidability either on the model or on the query language
Complexity Results $W \models \varphi$

- **LTL-FO**
  - PSpace (bounded arities)
  - ExpTime

- **CTL-FO**
  - ExpTime
  - co-NexpTime (states are propositional)

- **CTL*-FO**
  - 2ExpTime
  - ExpSpace (states are propositional)
Discussion

- **Fragile decidability results**
  - adding any schema constraints
  - “tiny” relaxation of the above restrictions
  - preserving full execution history, etc.
  produces undecidability

- **Comparison with DCDS (Calvanese et al.)**
  - allows external services via user input
  - allows arbitrary big databases
  - decidability for LTL (restricted CTL*) only
  - no schema constraints
7. Marc Spielmann, “Verification of relational transducers for electronic commerce”
9. Edgar F. Codd, “A Relational Model of Data for Large Shared Data Banks”
Thank you! Any Questions?