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*

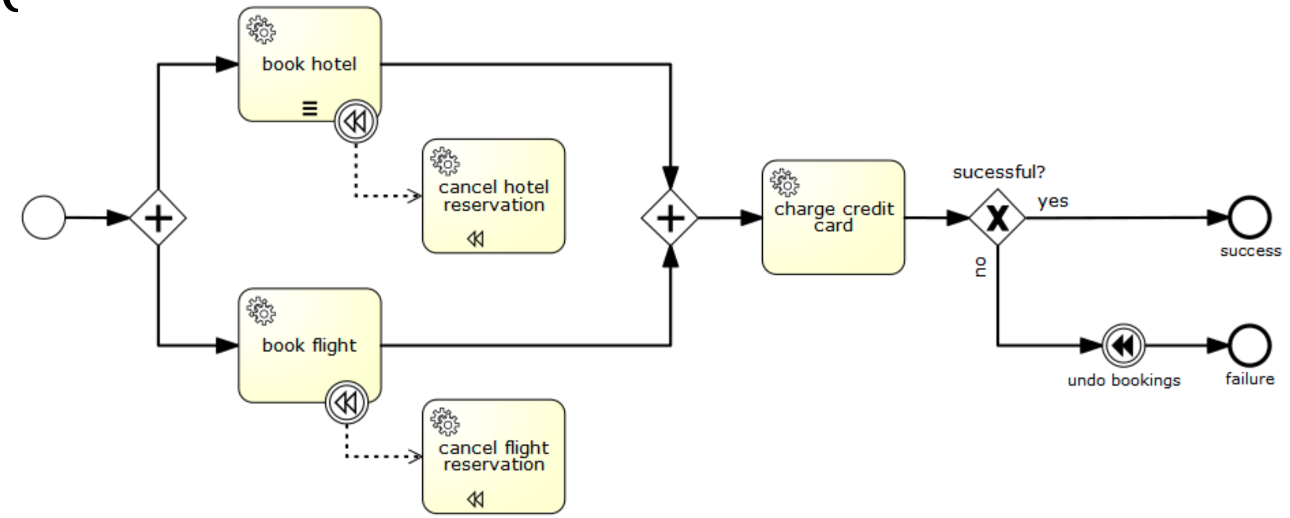
What are Business Processes?

- **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
    - **executable language** for specifying actions within business processes with web services
- **'05 BPMN**: Business Process Modeling Notation
    - **graphical** modeling language
    - 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

‘00 — **Relational Transducers,** Abiteboul and Vianu,2000

‘03 — **ASM transducers,** Spielmann,

‘03 — **Business Artifacts @ IBM**Nigam and Caswell

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

‘12 — **Data-Centric Dynamic Systems,** Calvanese et al, 12

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
  - enterprise-wide unique identity
  - self-describing content (Key, Value)
- Identity unchangeable, Unsplitable
  - multiple artifacts with same content but different id
- Manipulatable
  - update
  - copy from other artifacts
  - adds from any source (input, computation, ...)

Example: Burgershop

```
 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

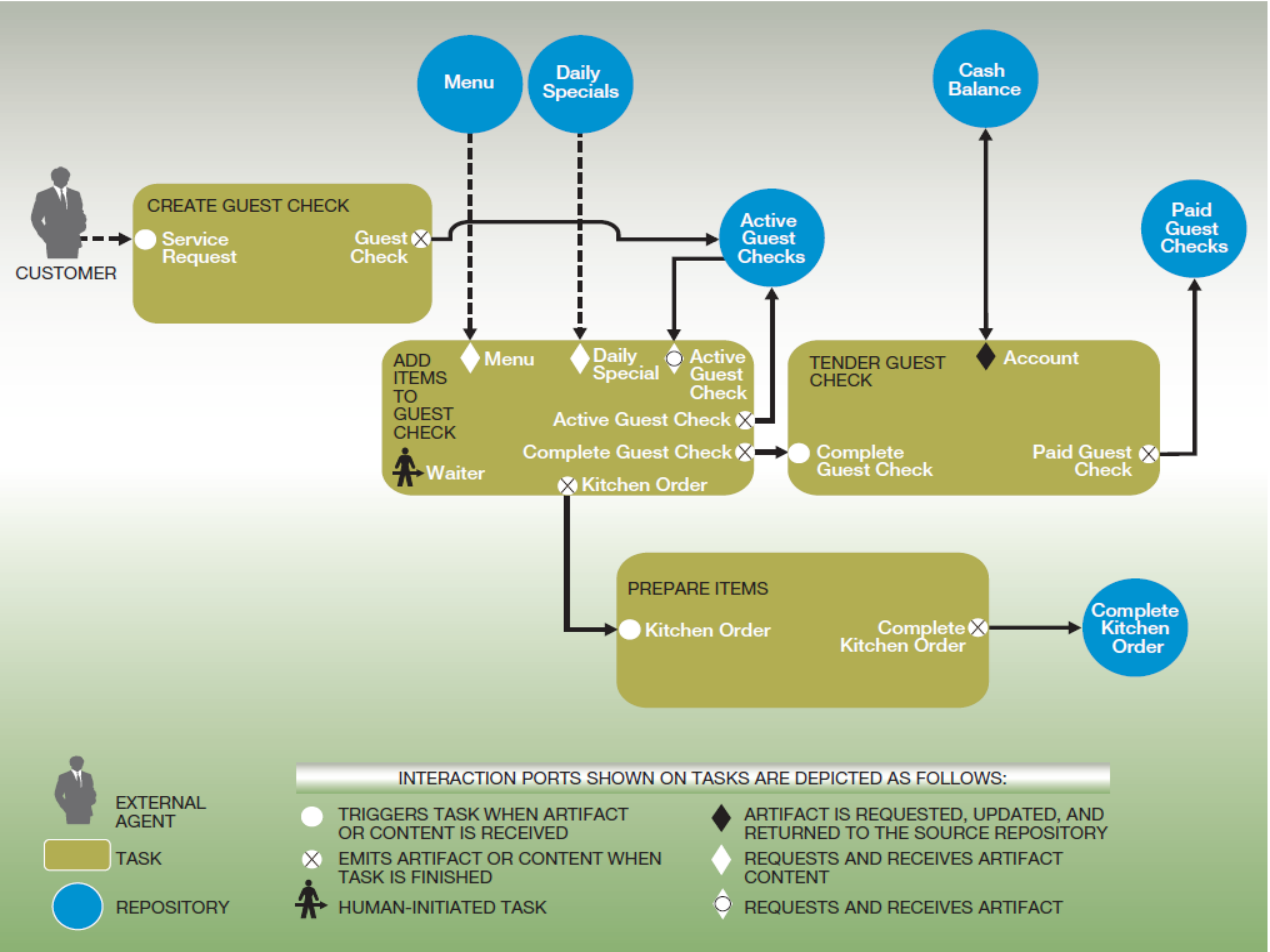
- each artifact has a lifecycle
- end-to-end processing
  - creation
  - completion
  - archiving
- Places
  - Tasks (changes to artifacts)
  - Repositories (artifacts await further processing)
- describes the operations of a business
  - Function: how to add/update information
  - Flow: transport across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


Menu
Daily Specials
Cash Balance
CUSTOMER
CREATE GUEST CHECK
Service Request
Guest Check
Active Guest Checks
Paid Guest Checks
ADD ITEMS TO GUEST CHECK
Menu
Daily Special
Active Guest Check
Active Guest Check
Complete Guest Check
Waiter
Kitchen Order
TENDER GUEST CHECK
Account
Complete Guest Check
Paid Guest Check
PREPARE ITEMS
Kitchen Order
Complete Kitchen Order
Complete Kitchen Order
EXTERNAL AGENT
TASK
REPOSITORY
INTERACTION PORTS SHOWN ON TASKS ARE DEPICTED AS FOLLOWS:
TRIGGERS TASK WHEN ARTIFACT OR CONTENT IS RECEIVED
EMITS ARTIFACT OR CONTENT WHEN TASK IS FINISHED
HUMAN-INITIATED TASK
ARTIFACT IS REQUESTED, UPDATED, AND RETURNED TO THE SOURCE REPOSITORY
REQUESTS AND RECEIVES ARTIFACT CONTENT
REQUESTS AND RECEIVES ARTIFACT

Modeling with Artifacts

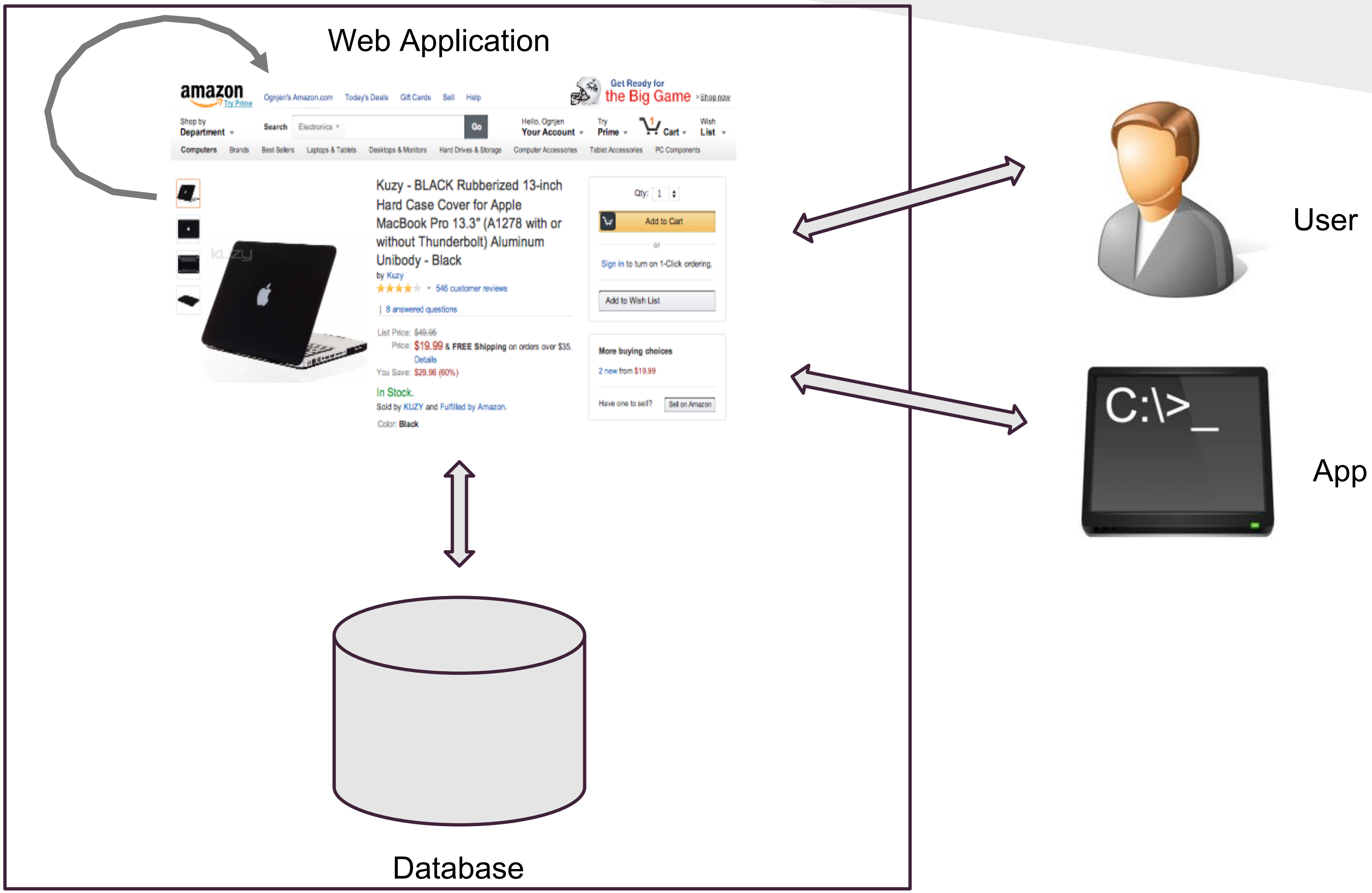
- Pick key artifacts, construct lifecycle
- Create a candidate list (all artifacts needed for key artifact)
- Repeat
  - Take artifact from candidate list
  - construct lifecycle
  - add newly emerging artifacts to candidate list

Specification and Verification of Data-driven Web Applications

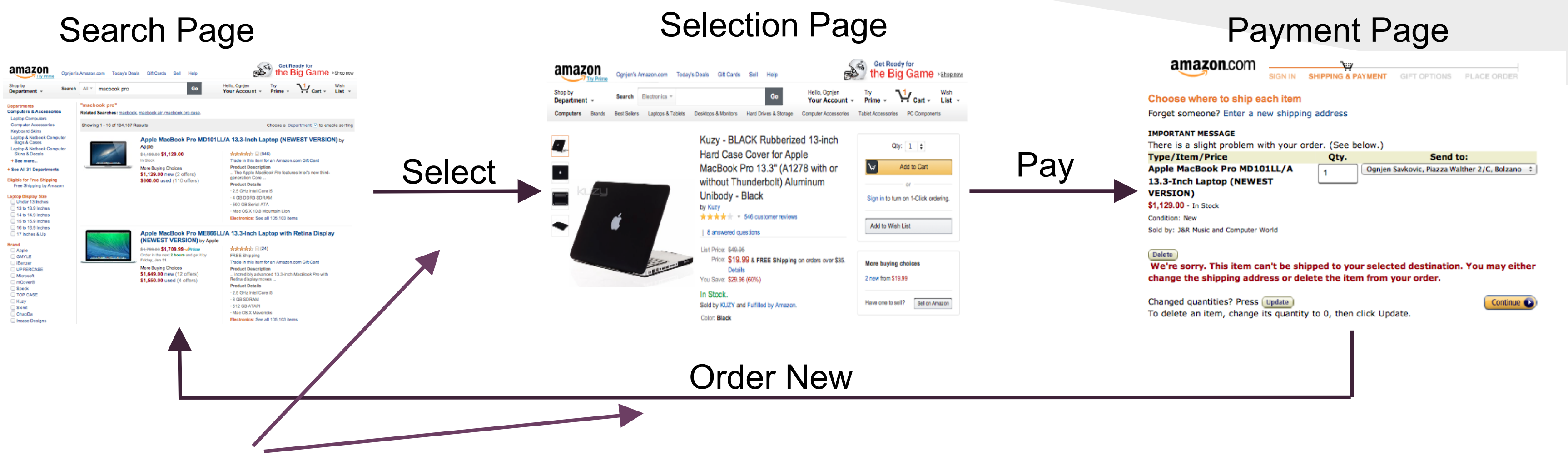
Alin Deutsch, Liying Sui and Victor Vianu

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

Definition: Web Application

A Web application $\mathcal{W}$ is a tuple $\langle \mathbf{D}, \mathbf{S}, \mathbf{I}, \mathbf{A}, \mathbf{W}, W_0, W_\epsilon \rangle$, where:

- $\mathbf{D}, \mathbf{S}, \mathbf{I}, \mathbf{A}$ are relational schemas:
  - Database schema
  - State schema
  - Input schema
  - Action schema
- $\mathbf{W}$ is a finite set of Web page schemas
- $W_0 \in \mathbf{W}$ is the home page schema, and $W_\epsilon \notin \mathbf{W}$ is the error page schema

Definition: Web App Schema

A Web page schema $W \in \mathbf{W}$ is a tuple $\langle \mathbf{I}_W, \mathbf{A}_W, \mathbf{T}_W, \mathcal{R}_W \rangle$, where:

- $\mathbf{I}_W \subseteq \mathbf{I}$, input set of the web page schema
- $\mathbf{A}_W \subseteq \mathbf{A}$, action set of the web page schema
- $\mathbf{T}_W \subseteq \mathbf{T}$, the set of target web pages
- $\mathcal{R}_W$, the set of rules containing:
  - · Input rules for each input relations $I \in \mathbf{I}_W$
  - · State rules for each state relations $S \in \mathbf{S}$
  - · Action rules for each action relations $A \in \mathbf{A}_W$
  - · Input rules for each input relations $V \in \mathbf{T}_W$

Input and State Rules

- **Input rules:** $Options_I(\tilde{x}) \leftarrow \varphi_{I,W(\tilde{x})}$ where $Options_I$ is a realtion of arity k, $\tilde{x}$ a k-tuple of distinct variables and $\varphi_{I,W}(\tilde{x})$ an FO formular over $\mathbf{D} \cup \mathbf{S} \cup \mathbf{Prev_I} \cup \mathbf{const}(\mathbf{I})$ with free $\tilde{x}$
  $\mathbf{D} \cup \mathbf{S} \cup \mathbf{Prev_I} \cup \mathbf{const}(\mathbf{I})$ 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 $\varphi^{\epsilon}_{S,W}(\tilde{x}),\ \epsilon \in \{+,-\}$ are FO formulars over schema $\mathbf{D} \ \cup \ \mathbf{S} \ \cup \ \mathbf{Prev_I} \ \cup \ \mathbf{const}(\mathbf{I}) \ \cup \ \mathbf{I}_W,$ 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 $\mathbf{D} \cup \mathbf{S} \cup \mathbf{Prev_I} \cup \mathbf{const}(\mathbf{I}) \cup \mathbf{I}_W$ with free variables $\tilde{x}$
- **Target rules** $V \leftarrow \varphi_{V,W}$ where $\varphi_{V,W}$ is an FO sentence over schema $\mathbf{D} \cup \mathbf{S} \cup \mathbf{Prev_I} \cup \mathbf{const}(\mathbf{I}) \cup \mathbf{I}_W$

$W_\epsilon = \langle \emptyset, \emptyset\{W_\epsilon\}, R_W \rangle$ where R_{W_ϵ} consists of the rule $W_\epsilon \leftarrow \ true$.

Semantics

A run of a Web app W over a database $\mathbf{D}$ 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 **h**ome **p**age
- **RP** - the new user **r**egistration **p**age
- **CP** - the **c**ustomer **p**age
- **AP** - the **a**dministration **p**age
- **LSP** - the **l**aptop **s**earch **p**age
- **PIP** - the **p**roduct **i**tem page, products returned by search
- **CC** - the **c**art **c**ontent
- **MP** - the error **m**essage **p**age

Example: Home Page

Page HP

Inputs $\mathbf{I}_{HP}$: $\mathtt{name}, \mathtt{password}, \mathtt{button}(\mathtt{x})$

InputRules :

$\text{Options}_{\mathtt{button}}(x) \leftarrow x = \text{``}login\text{''} \vee x = \text{``}register\text{''} \vee x = \text{``}clear\text{''}$

StateRules :

$\text{error}(\text{x}) \leftarrow \neg\underline{\mathtt{user}}(\mathtt{name}, \mathtt{password}) \wedge \mathtt{button}(\text{``login''}) \wedge x = \text{``}failed\ login\text{''}$

TargetWebPages $\mathbf{T}_{HP}$: HP, RP, CP, AP, MP

TargetRules :

$HP \leftarrow \mathtt{button}(\text{``clear''})$

$RP \leftarrow \mathtt{button}(\text{``register''})$

$CP \leftarrow \underline{\mathtt{user}}(\mathtt{name}, \mathtt{password}) \wedge \mathtt{button}(\text{``login''}) \wedge \mathtt{name} \neq (\text{``Admin''})$

$AP \leftarrow \underline{\mathtt{user}}(\mathtt{name}, \mathtt{password}) \wedge \mathtt{button}(\text{``login''}) \wedge \mathtt{name} = (\text{``Admin''})$

$MP \leftarrow \neg\underline{\mathtt{user}}(\mathtt{name}, \mathtt{password}) \wedge \mathtt{button}(\text{``login''})$

EndPage HP

Example: Laptop Search Page

Page LSP

Inputs $\mathbf{I}_{LSP}$: $\texttt{laptopsearchpage}(ram,\ hd,\ display), \texttt{button}(x)$

InputRules :

$\text{Options}_{\texttt{button}}(x) \leftarrow x = \text{“}search\text{”} \vee x = \text{“}viewcart\text{”} \vee$
$x = \text{“}logout\text{”}$

$\text{Options}_{\texttt{laptopsearch}}(r, h, d) \leftarrow \texttt{cirteria}(\text{“}laptop\text{”}, \text{“}ram\text{”}, r)$
$\wedge\, \texttt{cirteria}(\text{“}laptop\text{”}, \text{“}hdd\text{”}, h) \wedge \texttt{cirteria}(\text{“}laptop\text{”}, \text{“}display\text{”}, d)$

StateRules :

$\text{userchoise}(\text{r}, \text{h}, \text{d}) \leftarrow \texttt{laptopsearch}(r, h, d) \wedge \texttt{button}(\text{“}search\text{”})$

TargetWebPages $\mathbf{T}_{HP}$: HP, PIP, CC

TargetRules :

$HP \leftarrow \texttt{button}(\text{“logout”})$

$PIP \leftarrow \exists r \exists h \exists d\ \texttt{laptopsearch}(r, h, d) \wedge \texttt{button}(\text{“}search\text{”})$

$CC \leftarrow \texttt{button}(\text{“}view\ cart\text{”})$

EndPage LSP

Verification Language

- Verification of temporal aspects of web application
  - 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"

$$\forall pid, price\ [\xi(pid, price)]\ \mathbf{B}\ \neg(\overline{conf}(\mathtt{name}, price) \wedge \overline{ship}(\mathtt{name}, pid))]$$

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

$$\begin{aligned} PP\ & \wedge \mathtt{pay}(price) \wedge \mathtt{button}(\text{"authorized payment"}) \\ & \wedge \mathtt{pick}(pid, price) \\ & \wedge \exists pname\ \mathtt{catalog}(pid, price, pname) \end{aligned}$$

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"

$\forall pid \forall price\, A\mathbf{G}(\xi(pid, price) \rightarrow A((E\mathbf{F}\overline{cancel}(\texttt{name}, pid)\mathbf{U}(\overline{ship}(\texttt{name}, pid)))$
where $\xi(pid, price)$ is the formula
$PP \wedge \texttt{pay}(price) \wedge \texttt{button}(\text{``authorized payment''}) \wedge pick(pid, price) \wedge \underline{\texttt{prod\_price}}(pid, price)$

Undecidability

- Given Web App W and Temporal FO φ we want to check whether W |= φ

- Undecidability follows immediately :)
  - φ an FO sentence over D
  - action rule A ← φ, where A is a proposition
  - φ is finitely satisfiable iff A |= ¬G¬A
  - Trakhtenbrot's theorem: finite satisfiability of FO sentences is undecidable

Gaining Decidability

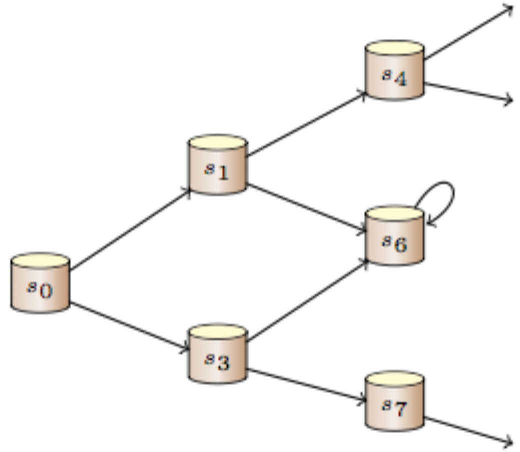
- To gain decidability **restrict** FO formulas to
  - "input-bounded" quantification (restricted FO)
  - all state atoms are ground

- 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 |= φ

- 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

References

1. Ruth Sara Aguilar Saven,”Business process modelling: Review and framework”
2. Van der Aals, W. “Process Mining. Discovery, Conformance and Enhancement of Business Processes”
3. Calvanese, D. , Montali, M., De Giacomo, G., “Foundations of Data-Aware Process Analysis: A Database Theory Perspective”
4. Cohn, D., Hull, R., “Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes”
5. Nigam, A., Caswell, N.S., “Business artifacts: An approach to operational specification”
6. Serge Abiteboul, Victor Vianu, Brad Fordham and Yelena Yesha, “Relational Transducers for Electronic Commerce”
7. Marc Spielmann, “Verification of relational transducers for electronic commerce”
8. Alin Deutsch, Liying Sui and Victor Vianu, “Specification and Verification of Data-driven Web Applications”
9. Edgar F. Codd, “A Relational Model of Data for Large Shared Data Banks”

Thank you! Any Questions?