Process-Algebra Based Query Workflows

T. Hornung[1], W. May[2], G. Lausen[1]

Universität Freiburg,
{hornungt|lausen}@informatik.uni-freiburg.de

Universität Göttingen,
may@informatik.uni-goettingen.de

10.06.2009

Motivating Example: Travel Planning

- *Query:* What is the cheapest connection from Freiburg to Amsterdam?
- *Solution:*
  - Consider nearest airports
  - Query train connections from start to airports
  - Query flights from airports to destination
  - ...

Motivating Example: Travel Planning

- *Query:* What is the cheapest connection from Freiburg to Amsterdam?
- *Solution:*
  - Consider nearest airports
  - Query train connections from start to airports
  - Query flights from airports to destination
  - ...

→ Design a *Query Workflow* to find connections *automatically*

The MARS Framework

- Modular Active Rules for the Semantic Web
- Provides an open framework for active rules and for processes
- Supports *heterogeneous event, query,* and *action* languages:
  - XQuery, SPARQL
  - *WSQL* for querying Web Services
  - *DWQL* for querying Deep Web sources
- *Data model:* Relational dataflow

  Set of tuples of variable bindings R, i.e., every tuple is of the form $t = \{v_1/x_1, \ldots, v_n/x_n\}$ with $v_1, \ldots, v_n$ variables and $x_1, \ldots, x_n$ elements of the underlying domain

Data Model: Example

Consider the query:

$$\text{Query}((dt, at, pr) \leftarrow \text{getTrainConnection}(start, dest, date))$$

which returns the departure time (dt), arrival time (at), and price (pr) for a train connection from *start* to *dest* on *date*:

$$\begin{aligned}
&\text{getTrainConnection}(\\
&\{\{start/\text{"FR"}, dest/\text{"BS"}, date/\text{"10.06.2009"}\},\\
&\ \{start/\text{"FR"}, dest/\text{"F"}, date/\text{"10.06.2009"}\}\}) =\\
&\qquad \{\{start/\text{"FR"}, dest/\text{"BS"}, date/\text{"10.06.2009"},\\
&\qquad\ \ dt/\text{"10:02"}, at/\text{"10:47"}, pr/\text{"21.80"}\},\\
&\qquad \ldots,\\
&\qquad \{start/\text{"FR"}, dest/\text{"F"}, date/\text{"10.06.2009"},\\
&\qquad\ \ dt/\text{"09:57"}, at/\text{"12:43"}, pr/\text{"61.00"}\}\}
\end{aligned}$$

Calculus of Communicating Systems (CCS)

- Process Algebra
- Express processes over atomic constituents by:
  - Sequence
  - Alternative
  - Concurrent
  - Recursion
- Process state is encoded implicitly in the process itself

Calculus of Communicating Systems (CCS)

- Process Algebra
- Express processes over atomic constituents by:
  - Sequence
  - Alternative
  - Concurrent
  - Recursion
- Process state is encoded implicitly in the process itself

→ RelCCS: Extension of CCS with relational dataflow

Atomic Constituents of RelCCS

Formal Semantics of Process Execution

$[\![\ ]\!] : \mathcal{P} \times 2^{\text{Tuples(Var)}} \rightarrow 2^{\text{Tuples(Var)}}$

- *Actions:* $[\![\text{Action}(a)[R]]\!] := R$, plus external side effects of a
- *Queries:* $[\![\text{Query}(q)[R]]\!] := \bigcup_{t \in R} [\![\text{Query}(q)[t]]\!] = R \bowtie q_0$
- *Tests:* $[\![\text{Test}(c)[R]]\!] = \sigma[c](R)$, like SQL's
  `SELECT * FROM R WHERE` c
- *Events:* Similar to queries. On occurrence of an event, matching tuples proceed through the process.

Selected Operators of RelCCS

- *Sequence:* $[\![\mathrm{Seq}(P, Q)[R]]\!] := [\![Q[[\![P[R]]\!]]]\!]$
- *Union:* Each branch is started with R.
  $[\![\mathrm{Union}(P_1, \ldots, P_n)[R]]\!] = [\![P_1[R]]\!] \cup \ldots \cup [\![P_n[R]]\!]$
- *Concurrent:* Each branch is started with R.
  $[\![\mathrm{Concurrent}(P_1, \ldots, P_n)[R]]\!] := [\![P_1[R]]\!] \bowtie \ldots \bowtie [\![P_n[R]]\!]$

Application Scenario: Travel Planning

process input: $(start, dest, date)$
Sequence(
  Query($rd \leftarrow$ distance($start, dest$)),
  Union(
    Sequence(Test($rd < 800$), # consider to go by train
      Query($(dt, at, pr) \leftarrow$ getTrainConnection($start, dest, date$)),
    Sequence(Test($rd \geq 400$), # consider also to use flights
      Query($ap \leftarrow$ getAirports()),
      Query($dist \leftarrow$ distance($start, ap$)),
      TopK(10,100,null,*dist*,xsd:decimal,asc,false),
      Query($(dt, at, pr) \leftarrow$ getTrainConnection($start, ap, date$)),
      UseDefinition(exploreTC[ap]))))

Definition(exploreTC[local: ap, ...]) := ...
postcondition: reached, either by train or train+flight$^+$, or
train+flight$^+$+train

Related Work

- Petri Nets
  - Graphical formalism with concise formal semantics
  - Support for formal analysis and verification techniques
- Yet Another Workflow Language (YAWL)
  - Based on exhaustive analysis of workflow patterns
  - Roots in Petri Nets
- WS-BPEL
  - Block-oriented process language, similar to procedural programming languages
  - Dataflow described by variables, that can be made set-oriented by referencing external database tables

Conclusion

- RelCCS: Extension of CCS with relational dataflow
  - Both the primitives for control structures and data flow are on the same level of the language
  - Embedding in MARS framework provides tight integration, e. g. RelCCS fragments can be used as action parts in MARS' active rules
  - Comfortable embedding of algorithmic Web Services (Configurable Graph Datatype)
- *Future Work:* Automatic derivation of processes and patterns from ontologies

Prototype Implementation

RelCCS is implemented in a prototype which can be found with sample processes and further documentation at http://www.semwebtech.org/mars/frontend/ → run CCS Process.

??? Questions ???

Take-Home Message

Often it is easier to design the process *how* to solve a problem than stating a single query