Reasoning in and for the Semantic Web

Wolfgang May

Institut für Informatik, Universität Göttingen, Germany

Workshop “Grundlagen von Datenbanken”, Wörlitz, Germany, May 20, 2005

Motivation and Overview

The goal of the Semantic Web is to have a semantic level access on the Web (including semantic-based query functionality), or at least to fragments of the Web. Semantic Web research combines (and requires) a broad spectrum of areas in computer science, from underlying technical issues, data management, knowledge representation, software engineering, communication technology, user interfaces/personalization up to logical formalisms and reasoning. Additionally, interdisciplinary research is done wrt. many different application areas, e.g., biology, business, linguistics.

REWERSE (REasoning on the WEb with Rules and SEmantics; http://rewerse.net) is a research “Network of Excellence” (NoE) funded by the EU Commission within the “6th Framework Programme” (FP6).

This talk gives a short introduction to the Semantic Web and an overview how REWERSE combines research areas from practical and theoretical computer science; here especially some issues relevant for database researchers, and especially in our workpackage are discussed.

References to several overview articles that have been prepared in the first year of REWERSE can be found in the paper.

Overview

1. Introduction: Semantic Web
2. Overview: concepts, issues, ...
3. Example: the area where our Workpackage is working on: Evolution and Reactivity

Querying the “the (HTML) Web” – until ∼ 200

- Web intended for browsing

```
<table>
  <tr> <td>Name</td> <td>Germany</td> </tr>
  <tr> <td>Code</td> <td>D</td> </tr>
  <tr> <td>Area </td> <td>365910</td> </tr>
    :
</table>
```

- wrapping HTML and form-based interfaces
- accessing multiple Web pages
- proprietary research solutions for “semistructured data” (e.g., Tsimmis, F-Logic, Lixto)
  - programming wrappers (HTML to internal format) and mediators (data integration in internal format)

Querying the “the HTML+XML Web”

Currently – Browsing + Querying

- parallel “Webs”: HTML Web and XML Web
- XML data accessible via Web

```
<Country code=“D”>
  <Name>Germany</Name>
  <Area>365910</Area>
    :
</Country>
```

- standardized Query Language: XQuery/XPath;
  language design used experiences from the network model, SQL, OQL, F-Logic, Lorel etc. ...
- different sources use different structures and different “names”

⇒ still manual programming of integration/mediators

Querying “the Web”?

... this is actually “Querying XML on the Web”
(+ integration of data from the Web):

- explicit addressing of a certain Web resource that holds the information
- restricted use of inter-source references
- integration problems when combining resources that use different *schemata*
- depends on the *structure* of the provided data

$\Rightarrow$ not “the Web”, and not semantics-oriented.

Querying “the Web”!

... continue the idea of today’s portals:

- independent from the actual location of information
- combination of information by “the Web” (not by the user) (e.g. portals)
- information sources must support this (cooperation)
- semantics-based, not syntax/data-structure-based querying

The Semantic Web: RDF

- the “Web” structure of documents/data provides background infrastructure to distributed data (whose actual model is independent from their location)
- location of *services* still given by URLs

RDF: Resource Description Format

- Triples: (subject (resource), predicate, object (resource))
- Resources: “objects” of interest, described by URIs (uniform resource identifiers)
- RDF: node- and edge-labeled graph data model (cf. OEM)
- RDF *can* be *represented* by XML data
- distribution of this (mostly XML) data independent from the described objects

⇒ distributed, strongly intertwined, but very loosely coupled database

RDF

Example:

```
<rdf:Description rdf:about="http://www.deutschland.de">
  <rdf:type rdf:resource= "http://geo.org/ontology/#country"/>
  <name>Germany</name>
  <code>D</code>
  <has_city>
    <rdf:Description rdf:about= "http://www.berlin.de">
      <rdf:type rdf:resource= "http://geo.org/ontology/#city"/>
      <name>Berlin</name>
    </rdf:Description>
  </has_city>
  <has_capital rdf:resource="http://www.berlin.de"/>
</rdf:Description>
```

- there can be different RDF files that also describe Germany and Berlin, using the same URIs.

The Semantic Web: Data Model and Semantic

... and now we can talk about “querying the Web”:

- global data model:
  - RDF format
  - RDF Schema
    defines the notions that are used with <rdf:type>,
    and the notions that are used as property names
  - OWL (Web Ontology Language (+Semantics))
- note: not the XML representation is relevant, but only the RDF triples that are described by it!
- no integration necessary if sources agree on the used ontologies (names+URIs)

The Semantic Web: Requirements

- communication for actual distributed query answering
  (+ mapping from local information to global format)
- global model, notions of (restricted) consistency
- global strategies for propagation of information and changes.

Contributing Subdisciplines and Issues

- communication: protocols, security, infrastructure, ... (peer-to-peer, publish-subscribe, continuous queries, ...)
- individual nodes: modeling, data model, metadata, query+update languages (design+implementation), reasoning (e.g., OWL)
- Web Services: specification of target-driven (distributed) behavior and communication
- Information integration: interoperability, data integration, inter-ontology mappings; languages, reasoning
- “social”: community building, trust, security, recommender systems
- User interfaces: personalization, adaptivity
- Applications: modeling, validation of techniques, feedback

The REWERSE NoE

- REasoning on the WEb with Rules and SEmantics; http://rewerse.net)
- research “Network of Excellence” (NoE) funded by the EU Commission within the “6th Framework Programme” (FP6).
- 27 universities and companies participate (about 120 people)
- Coordinator: Prof. Dr. François Bry, LMU Munich

Workpackages

- Foundations: Rule Markup, Policies, Composition & Typing, Query, Evolution,
- Applications: Time & Location, Bioinformatics, Personalisation
- Activities: Education and Training, Technology Transfer

Individual (Passive) Semantic Web Node

- local state, fully controlled by the node
- [optional: local behavior; see later]
- stored somehow: relational, XML, RDF databases
- local knowledge: KR model, notion of integrity, logic
  Description Logics, F-Logic, OWL
- query/data manipulation languages:
  - database level, logical level
- mapping? – logics, languages, query rewriting, query containment, implementation
- For this *local* state, a node should guarantee consistency

Local-Global View: Knowledge States

A Web node has not only its own data, but also “sees” other nodes:

- infrastructure: heterogeneity of data models
  - goal of the semantic level modeling
  - agreements on ontologies (application-dependent)
  - agreement on languages (e.g., RDF/S, OWL)

  in some sense solved (federated databases):
  in a closed world no problem $\rightarrow$ current portals
- how to deal with inconsistencies?
  accept them and use appropriate model/logics,
  reification/annotated statements (RDF), disjunctive logics
  or try to fix them $\rightarrow$ evolution of the SW
- sufficient e.g. in bioinformatics: sources are known
- insufficient: e.g. travel planning

Local-Global View (Cont'd)

Non-closed world

- incomplete view of a part of the Web
  - how to deal with incompleteness?
    different kinds of negation
- how to extend this view?
  - find appropriate nodes
    - information brokers, recommender systems
    - negotiation, trust
  - ontology querying and mapping
- static (model theory) vs. dynamic (query answering in restricted time)
- different kinds of logics, belief revision etc.

Languages: Querying

- an external query language
- powerful internal languages
  - mapping from actual data sources (XML) to higher level
- query answering consists of
  - finding appropriate data (using metadata + communication)
  - remote queries against certain sources
  - remote queries against “the Web”
  - deriving information, reasoning (intensional data)

⇒ Modular family of (sub)languages

XQuery too weak (metadata handling); DL, F-Logic etc., 2nd order

typing, formal semantics, interoperability, composability

not only between two given languages, but also on the meta level

Languages: Updating Data on the Web

Usually, query languages are directly extended to update languages (e.g. SQL, XQuery + Updates)

- updates of local data
- remote updates?
  - explicit statements in XQuery+Updates against a certain remote source
  - transaction functionality
    distributed, long, hierarchical transactions
- remote updates by messages/method calls – mapped to local updates
- intensional updates
  ... behavior of “the Web” as a whole

REWERSE I5: Evolution

Local Evolution of Web Nodes

- Web nodes with local behavior
- react on
  - local updates
  - incoming messages
  - temporal events
  - possibly poll/query other sources

$\Rightarrow$ Trigger-like
update/message + condition $\leadsto$
update (possibly including a remote query)

Global Evolution

Semantic Web as a whole as a network of *communicating* nodes

Dependencies between different Web nodes require to propagate changes on a node of the Web:

- view-like with explicit reference to other sources
  - \+ always uses the current state
  - \- requires permanent availability/connectivity
  - \- temporal overhead
- materialize the used information
  - \+ fast, robust, independent
  - \- potentially uses outdated information
- view maintenance strategies (web-wide, distributed)

⇒ specify and implement propagation by communication/propagation strategies

Propagation of Changes

Information dependencies induce communication paths:

- direct communication: subscribe – *push*
  based on registration; requires activity by provider
- direct communication: polling – *pull*
  regularly evaluate remote query
  – yields high load on “important” sources
  – outdated information between intervals
- + mapping into local data, view maintenance

Indirect Communication

Communication via intermediate services:

- indirect communication: publish/subscribe – *push/push*
  + requires (less) activity by provider
- indirect communication: continuous queries – *pull/push*
  register query at a cq service
  + acceptable load also for “important” sources
  + shorter intervals possible

Intermediate services can add functionality:

- data integration from several services
- checking query containment
- caching
- acting as information brokers (possibly specialized to an application area)

(Re)Activity & Evolution

- intended basic paradigm: reactivity
  - communication
  - specification and implementation of local behavior

$\Rightarrow$ homogeneous, modular framework

ECA Rules

"On Event check Condition and then do Action"

- sublanguages for specifying Events, Conditions, Actions
- Active Databases

Events

- local events (updates on the local knowledge)
- communication events (wrapped as messages)
  - explicit queries (XML: XQuery etc.; Semantic Web: RQL etc)
  - answers (in XML/RDF)
  - other messages

    $\Rightarrow$ markup language for general messages (SOAP-like)

Events on the Semantical Level

Declarative extension for evolution of “the Web”:
global application-level events “somewhere in the Web”

- “on change of VAT do ...”
- “if there is a flight FRA to LIS under 100E”
- local event (update) to a certain Web source
  (but: application not actually interested where it happens)
- different updates can raise the event
  “intensional” events

⇒ requires detection/communication strategies

Event Algebras

- specifying composite events as terms over atomic events.
- well-investigated in Active Databases (SNOOP etc.)

$$(E_1 \nabla E_2)(t) \;:\Leftrightarrow\; E_1(t) \vee E_2(t)\ ,$$
$$(E_1 ; E_2)(t) \;:\Leftrightarrow\; \exists t_1 \leq t : E_1(t_1) \wedge E_2(t)\ ,$$

Advanced Operators

- “negated” events, “cumulative” events

“Cumulative aperiodic event”

$$A^*(E_1, E_2, E_3)(t) \;:\Leftrightarrow\; \exists t_1 \leq t : E_1(t_1) \wedge E_3(t)$$

“if E_1 occurs, then for each occurrence of an instance of E_2, collect its parameters, and when E_3 occurs, report all collected parameters”.

Event Detection

- Local events, incoming messages, temporal events: easy
- composite events of event algebras:
  incrementally on-the-fly: automata/graphs, temporal formulas,
  extended with variable binding (parameters)
- intensional events
  - detection by derivation from another event
  - translation to a query

Conditions

= Queries

- local
- distributed/remote at a certain node
- Semantic Web-level

Actions

- updates of the local state:
  - facts
  - knowledge derivation rules
  - rules describing the behavior of a node
  - all kinds of knowledge are updatable
    local evolution
- calls of procedures/services
- sending messages
- transactions
  including queries against other sources
- intensional updates (e.g. in *business rules*)
  - “... then raise VAT by 1%”
- reasoning (about global effects, state etc.)

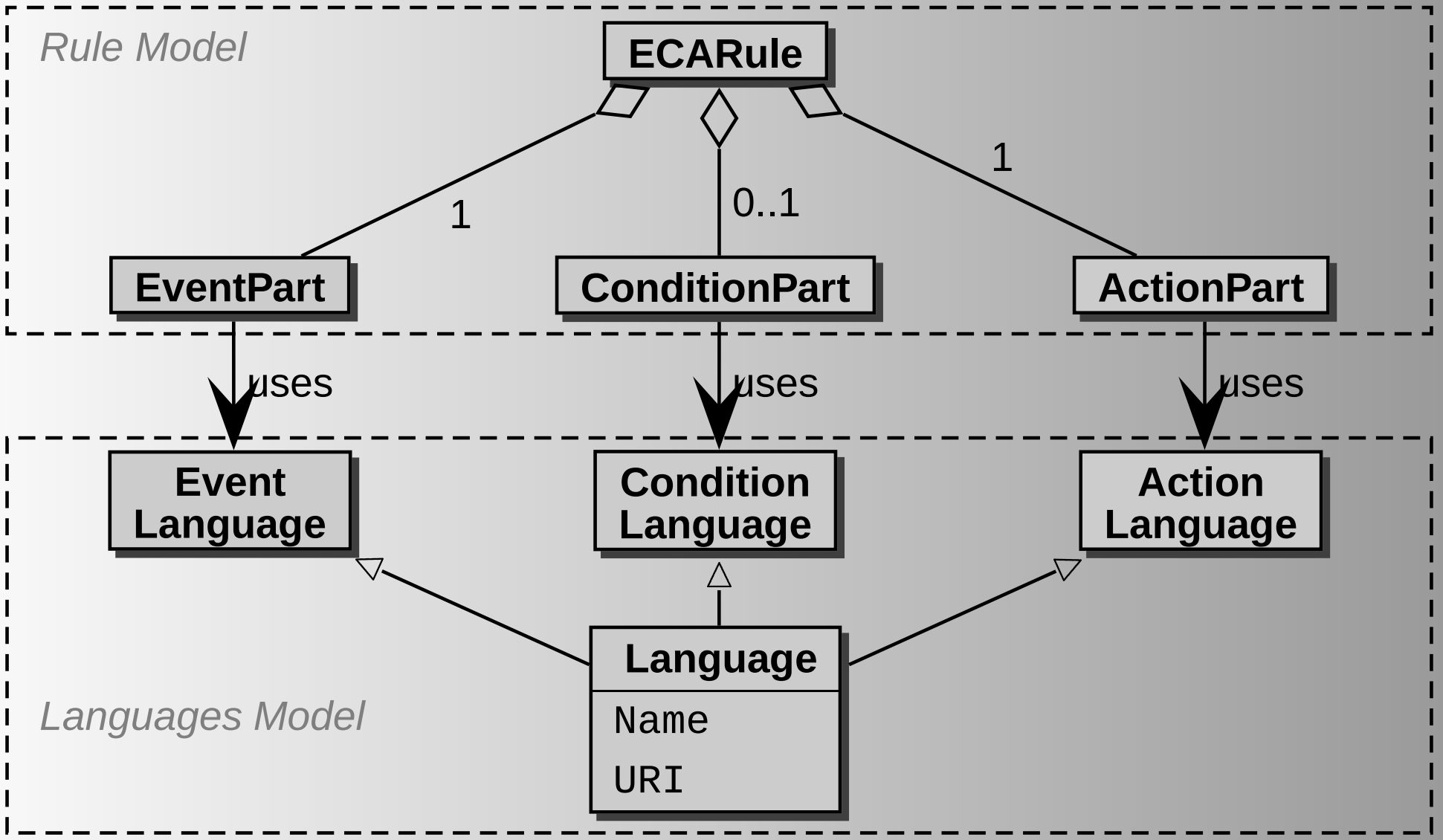
Diversity

common framework: ECA rules

- different levels of rules: basic triggers, simple rules, business rules
- rules over distributed data and distributed events with distributed actions
- intensional events and actions,
- rules using different E/C/A languages for the same application, probably on the same data
- nodes that are not able to execute rules

$\Rightarrow$ Modular concepts with Web-wide services

Modular ECA Concept: Rule Ontology



Rule Markup: ECA-ML

<**eca:rule** *rule-specific attributes*>
  declaration of variables
  <**eca:event** *identification of the language* >
    event specification
  <**/eca:event**>
  <**eca:condition** *identification of the language* >
    condition specification
  <**/eca:condition**>
  <**eca:action** *identification of the language* >
    action specification
  <**/eca:action**>
<**/eca:rule**>

- rule engine employs other services for E/C/A parts

Sublanguages: Event Markup

- Subelements contain a language identificator, and appropriate contents

```
<eca:rule>
   :
  <eca:event xmlns:snoop="snoop's URL">
    <snoop:sequence>
      <eca:atomic-event> atomic event spec </eca:atomic-event>
      <eca:atomic-event> atomic event spec </eca:atomic-event>
    </snoop:sequence>
  </eca:event>
   :
</eca:rule>
```

Service-Based Architecture

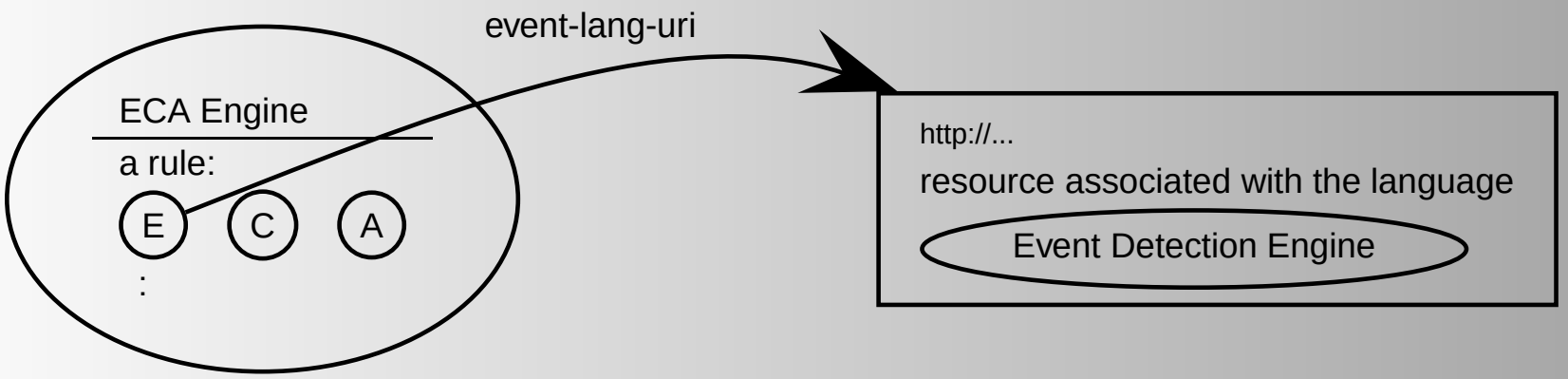
- Rule Execution Engine(s):
  nodes register their rules at the engines; everything else is done by the engine.
- Composite Event Detection Engines:
  implement detection of one or more event algebras
- analogous for condition and action parts

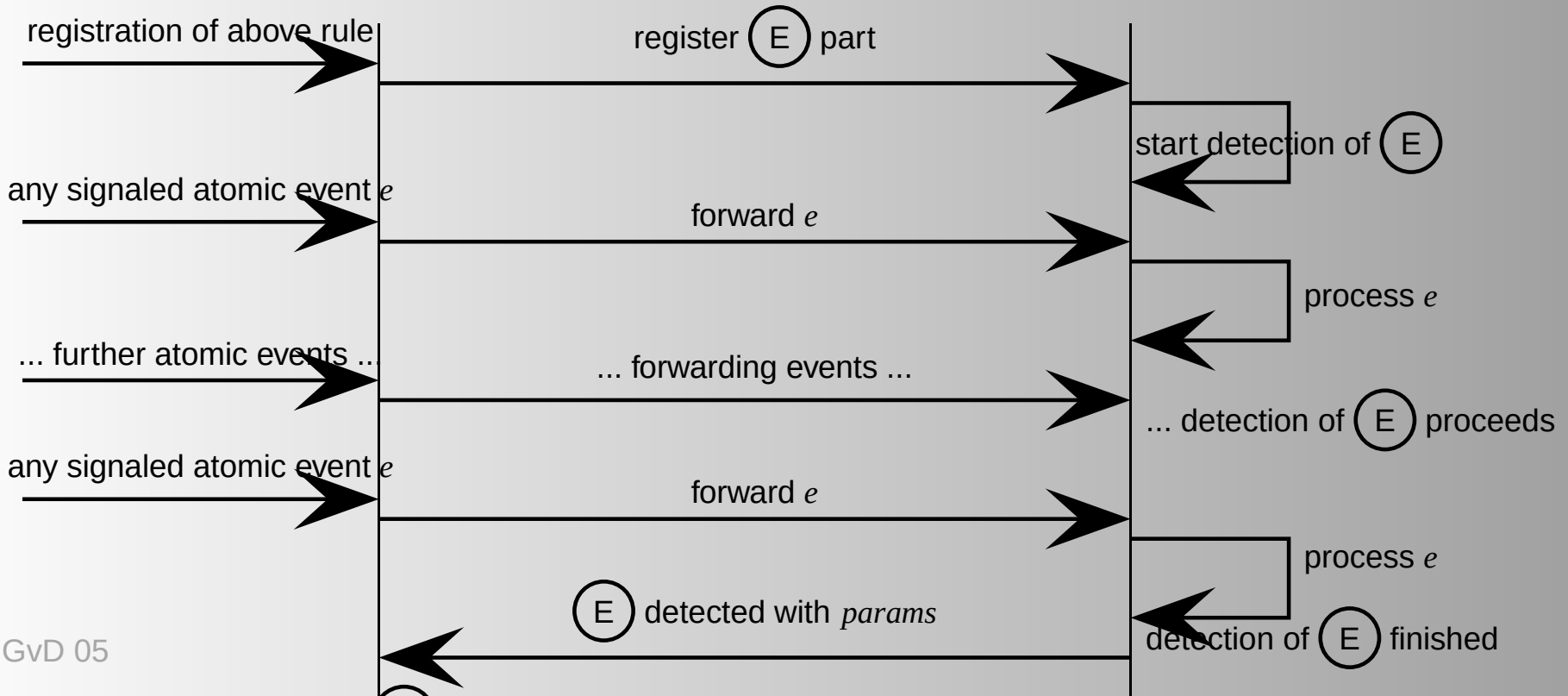
Communication of Events: Simple Pattern

Only the algorithmic part is outsourced:

- nodes that register a rule at a service must forward all known events to the service
- service that registers a composite event specification at a service must forward events to the service

Architecture



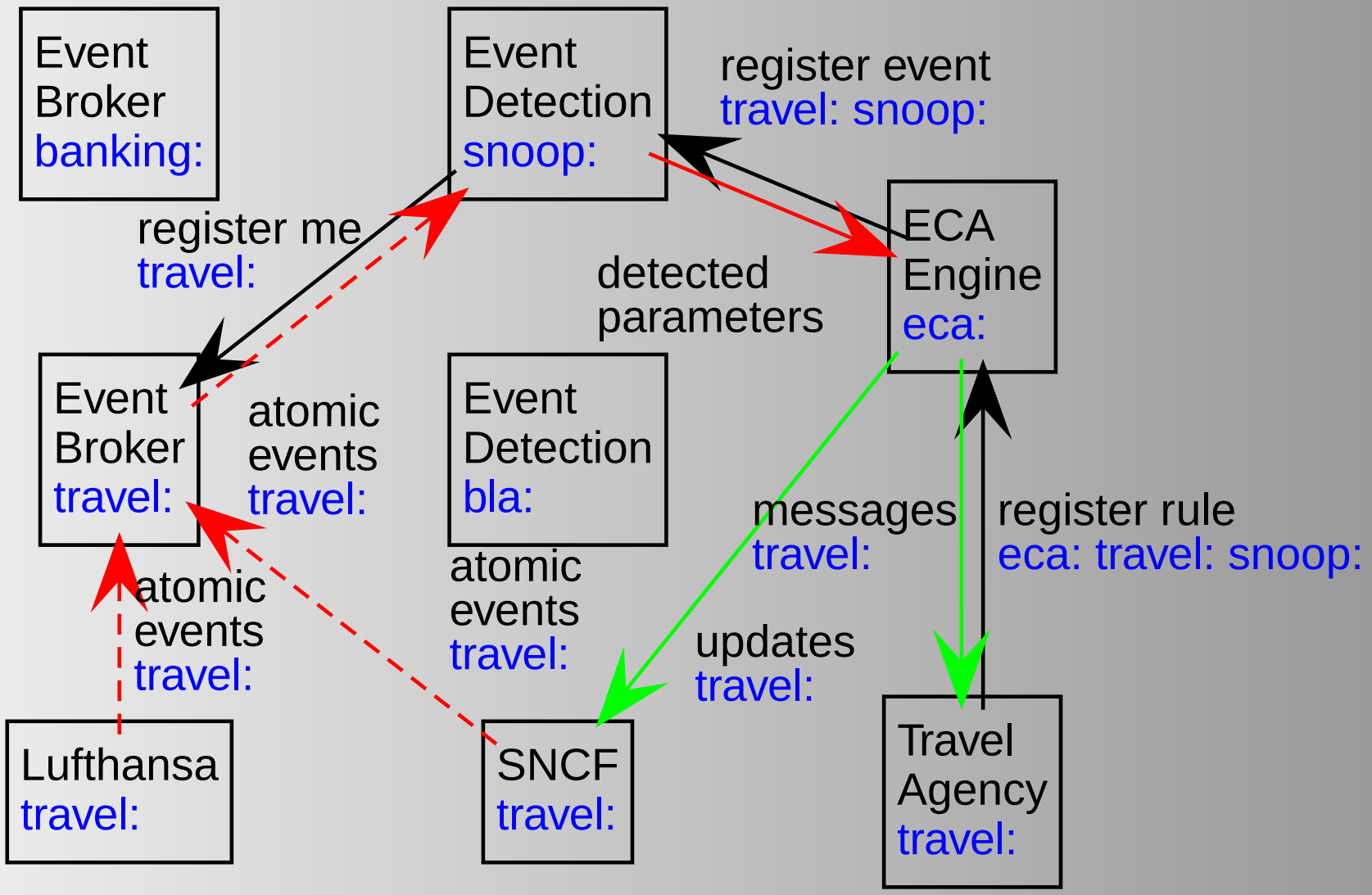


Advanced Architecture

Complete event detection is outsourced:

- composite event detection service is also responsible for detecting appropriate atomic events
  (e.g., specialized on a certain application area)
- Rule execution services
- Event broker services (application-specific)
- Algorithmic services (event detection, transactions)
- simple nodes that provide application-oriented functionality
  (e.g., travel agencies)

Architecture



Summary (Evolution and Reactivity)

- first: diversity looked like a problem
- now: diversity + unified, SW-based framework has many advantages
- languages of different expressiveness/complexity available
- functionality provided by specialized nodes
- sufficient if event detection services are informed about events
  communication can also be centered in some nodes

Summary

- combine subdisciplines of computer science
- reuse and adapt/extend known concepts
- learn from the problems of Artificial Intelligence in the 80s/90s