What Happened to Bob?
Semantic Data Mining of Context Histories

Michael Wessel, Marko Luther, Ralf Möller
Racer Systems • DOCOMO Euro-Labs • Uni Hamburg
A mobile community service
- 1300+ users in 60+ countries
- Connected to emerging Web 2.0 services

Your digital life recorder
- Facilitating context awareness on standard phones
- Integration of key Semantic Web technologies
What for?

- Real time context sharing
- Life blogging and widgets
- Automatic photo & video tagging
Today was the last day of my business trip to the ISWC’05 conference in Galway together with my colleague M. Luther. It was a cold and rainy day. In the afternoon I traveled back to Munich via Dublin by plane.
Situations & Situation Recognition

- Situations are vectors of attribute-value pairs (CA x CV) in the ABox
  \[
  \{ \text{sit} : \text{situation}, \quad (\text{sit}, \text{val}_1) : \text{CA}_1, \quad \text{val}_1 : \text{CV}_1, \ldots, \\
  (\text{sit}, \text{val}_n) : \text{CA}_n, \quad \text{val}_n : \text{CV}_n \} 
  \]
- + assertions for nearby persons, locations, social networks, ...
- reasoning required to recognize CA x CV occurrences

- Recognition with defined concepts and queries (rules)

\[
\text{business\_meeting} \equiv \geq 3 \ \text{near\_by\_colleague} \land \exists \text{at\_place\_office} \\
\text{ans}(x) \leftarrow \text{business\_meeting}(x), \quad \text{near\_by}(x, y), \text{near\_by}(y, z), \text{near\_by}(z, x), \\
\text{hates}(x, y), \text{hates}(y, z), \text{hates}(z, x).
\]
States, Events, Temporal Relations

- State = Situation + Temporal Information
  - Conceptually: linear discrete time model \((\mathbb{N}, <)\)
  - Various representation options for the \(<\) relation (below)

- Events are aggregates & intervals
  
  \[
  \text{simple\_event} \equiv \exists \text{start\_state}\cdot \text{state} \land \exists \text{end\_state}\cdot \text{state} \\
  \text{complex\_event} \equiv \text{event} \land \exists \text{has\_part}\cdot \text{event}
  \]

- Definition of complex events in terms of Allen relations
  
  - Allen relations computed from interval endpoints, e.g. a “stressful office day”

\[
\text{in\_office}(p_1), \text{meeting\_with\_boss}(p_2), \text{meeting\_with\_customer}(p_3), \\
\text{meets}(p_2, p_3), \text{during}(p_2, p_1), \text{during}(p_3, p_1)
\]
Realization of Event Recognizers

- Assuming events are already present as aggregates in the ABox, how to recognize them?
  - Defined concepts
    - Problems with relational expressivity for complex events
      - only tree-shaped temporal constraints expressible
      - Important event properties cannot be expressed (required for definitions of complex events!)
  - Queries and rules
    - High relational expressivity over the ABox
    - Universal closed-domain quantifier (SPARQL, SQL, nRQL)
Recognizing Homogeneous Events (1)

\[ \text{ans}(s_1, s_2) \leftarrow \text{state}(s_1), \text{state}(s_2), \text{future}(s_1, s_2), \]
\[ P(s_1), P(s_2), \]
\[ \backslash \pi(s_1)(\text{state}(s_0), \text{next}(s_0, s_1), P(s_0)), \]
\[ \backslash \pi(s_2)(\text{state}(s_3), \text{next}(s_2, s_3), P(s_3)), \]
\[ \backslash \pi(s_1, s_2)(\text{state}(s_3), \text{future}(s_1, s_3), \text{future}(s_3, s_2), \]
\[ \backslash P(s_3)) \]

(retrieve (?s1 ?s2)
  (and (?s1 state) (?s2 state) (?s1 ?s2 future)
    (?s1 P) (?s2 P)
    (neg (project-to (?s1) (and (?s0 state) (?s0 ?s1 next) (?so P)))
    (neg (project-to (?s2) (and (?s3 state) (?s2 ?s3 next) (?s3 P)))
    (neg (project-to (?s1 ?s2) (and (?s3 state) (?s1 ?s3 future) (?s3 ?s2 future)
      (neg (?s3 P)))))))

Recognizing Homogeneous Events (2)

\[ S_A = (\Delta^I, C^I, ..., R^I, ...), \text{ with} \]
\[ \Delta^I = \text{inds}(A), \]
\[ C^I = \{ i \mid i \in \text{inds}(A), A \vdash C(i) \}, \]
\[ R^I = \{ (i, j) \mid \text{inds}(A), A \vdash R(i, j) \}, \]

\[
\{ (s_1, s_2) \mid \exists s_1, s_2 : \text{state}(s_1) \wedge \text{state}(s_1) \wedge \text{future}(s_1, s_2) \wedge \\
P(s_1) \wedge P(s_2) \wedge \\
\neg \exists s_0 : \text{state}(s_0) \wedge \text{next}(s_0, s_1) \wedge P(s_0) \wedge \\
\neg \exists s_3 : \text{state}(s_3) \wedge \text{next}(s_2, s_3) \wedge P(s_3) \wedge \\
\neg \exists s_3 : \text{state}(s_3) \wedge \text{future}(s_1, s_3), \text{future}(s_3, s_2) \wedge \\
\neg P(s_3) \} \]
Where do Events Come From?

- Complex events can neither be reliably recognized nor correctly constructed with TBox axioms:

  \[
  \text{at\_home} \sqcap \exists\text{future}\text\_in\_office \sqsubseteq \exists\text{part\_of}\text\(). (\text{home\_2\_office\_event} \sqcap \exists\text{has\_part\_in\_office})
  \]

- Possible principle solutions with standard DLs
  - Programmatic pre-construction of (some? all?) events
    - try to recognize as many events as possible with defined concepts
    - (Some? All?) Allen relations can be precomputed
  - Dynamic construction of events with non-safe rules
    - Allen relations have to be computed dynamically for fresh events
Approach 1- Pre-constructed Events

- For a fixed number of states, all events and Allen relations between them are pre-constructed
  - Number of pre-constructible events is infinite, since complex events can have complex events as subevents, ...
  - Upper bound for non-recursive events can in principle be computed, but the number is very very large
  - (Complex) events are recognized rather than constructed
    \[
    \text{stressful\_office\_day}(x) \leftarrow \text{has\_part}(x, p_1), \text{has\_part}(x, p_2), \text{has\_part}(x, p_3), \\
    \text{in\_office}(p_1), \text{meeting\_with\_boss}(p_2), \\
    \text{meeting\_with\_customer}(p_3), \\
    \text{meets}(p_2, p_3), \text{during}(p_2, p_1), \text{during}(p_3, p_1)
    \]
  - Sometimes, defined concepts are sufficient
Approach 1- Pre-constructed Events

Complex Events on Level 2

Has part

Complex Events on Level 1

Simple Events

Start / end state

States
Approach 1 - Pre-constructed Events

- Complex Events on Level 2
- Complex Events on Level 1
- Simple Events
- States

Diagram showing relationships between states and events, with arrows indicating transitions and connections labeled "meets" and "next".
Approach 1 - Pre-constructed Events
Approach 1- Pre-constructed Events

Not feasible due to the enormous ABox sizes
(too many irrelevant events)
Approach 2 - Constructing Events

- All events are constructed dynamically
  - Non-safe & non-horn rules required, e.g. nRQL rules
    
    \[
    \text{stressful\_office\_day}\left(\text{new}(\text{sod}, p_1, p_2, p_3)\right),
    \text{has\_part}\left(\text{new}(\text{sod}, p_1, p_2, p_3), p_1\right),
    \text{has\_part}\left(\text{new}(\text{sod}, p_1, p_2, p_3), p_2\right),
    \text{has\_part}\left(\text{new}(\text{sod}, p_1, p_2, p_3), p_3\right)
    \]
    \[
    \leftarrow \text{in\_office}(p_1), \text{meeting\_with\_boss}(p_2),
    \text{meeting\_with\_customer}(p_3),
    \text{meets}(p_2, p_3), \text{during}(p_2, p_1), \text{during}(p_3, p_1)
    \]
  
  - Termination: make rules non-monotonic (acyclic definitions)
    - Construct only one $\text{stressful\_office\_day}$ per $p_1, p_2, p_3$
    - add NAF-negated conclusion to precondition
Approach 2 - Constructing Events
Approach 2 - Constructing Events
Approach 2 - Constructing Events

Complex Events on Level 1

Simple Events

States

Start / end state

Has part

meets

meets

next

next
Approach 2 - Constructing Events

Lots of rules that are very expensive too evaluate
Approach 3 - Combined Strategy

- All simple events which can be recognized without reasoning are pre-constructed

→ Keeps number of required rules small

- Leaves only the hard work (complex events and simple events that require reasoning) for the rule engine / reasoner
- Only for those homog., max., min. have to be verified
Approach 3 - Combined Strategy

One pre-constructed event per CA x CV constancy (homogeneous)

Simple events whose recognition requires reasoning or complex events are constructed dynamically
Computation of Allen Relations

- In Approach 2 and 3, efficient computation of Allen relations is crucial
  - Allen relations are defined in terms of endpoint / state relations
  - Options for representation of “<”:
    - explicit (“next” role assertions)
      - bigger Aboxes (index, no recomputation)
    - implicit (concrete domain or “data substrate”)
      - more complex queries (no index, recomputation, computation only on demand)
  - On the level of events:
    - “Implicit” vs. “explicit” Allen relations
Computation of Allen Relations

Allen Relations Per Second

- Implicit, concrete domain time attribute: 1.8 ARPS
- Explicit, next role assertions: 66 ARPS
- Implicit, “data substrate” time attribute: 170 ARPS
- Implicit, procedural solution: 600 ARPS

4225 Allen relations
From States to Events

- Conclusion: on the level of the states, relation representation doesn't matter too much (if CD avoided)

- This changes on the level of events, experiment:
  - Input: 142 individuals, 29 simple events, 131 relations
  - Output: 145 individuals, 29 simple events, 1 complex event, 5068 relations, 2025 Allen relations
  - Explicit Allen relations (role assertions): 73 seconds
  - Implicit Allen relations (definitions of event rules): hours

- Explanation
  - Query bodies get very complex, Allen relations are recomputed again and again
  - Top-down evaluation bad (bottom up / caching for Allen required)
“Special Day” Recognizer Rule

(prepare-abox-rule (and (?e1 in-region-is-home-event) (?e2 in-region-is-office-event)  
(or (and (?e3 business-lunch-event)  
     (?e2 ?e3 g-future)  
     (?e3 ?e4 g-future)))  
(and (?e3 nearby-is-supervisor-event)  
     (?e2 ?e4 g-future)  
     (?e1 ?e3 g-future)  
     (?e3 ?e5 g-future)))  
(?e4 in-region-is-office-event) (?e5 in-region-is-home-event)  
(?e1 ?s1 START-STATE) (?e2 in-region-is-office-event)  
(?e4 in-region-is-office-event) (?e5 in-region-is-home-event)  
(?e5 ?s2 END-STATE) (?e1 ?e2 g-future) (?e4 ?e5 g-future)  
(neg  
     (project-to (?s1 ?s2)  
     (and (?e special-work-day-event) (?e ?s1 START-STATE)  
         (?e ?s2 END-STATE))))))  
((instance (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) special-work-day-event)  
(related (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) ?s1 START-STATE)  
(related (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) ?s2 END-STATE)  
(related (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) ?e1 subevent)  
(related (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) ?e2 subevent)  
(related (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) ?e3 subevent)  
(related (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) ?e4 subevent)  
(related (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) ?e5 subevent))  
:abox default))
"Special Day" Expanded (DNF)

\[
\text{(union)}
\]
\[
\text{(?e1 in-region-is-home-event) (?e1 time-interval)}
\]
\[
\text{(?e3 business-lunch-event) (?e3 time-interval)}
\]
\[
\text{(?e3 ?INT-AN01-ano23 start-time-point)}
\]
\[
\text{(?INT-AN02-ano22 ?INT-AN01-ano23 tp-before-or-equal-tp)}
\]
\[
\text{(?e2 ?INT-AN02-ano22 end-time-point)}
\]
\[
\text{(?e2 in-region-is-office-event) (?e2 time-interval)}
\]
\[
\text{(?e3 ?INT-AN04-ano24 end-time-point)}
\]
\[
\text{(?INT-AN04-ano24 ?INT-AN03-ano25 tp-before-or-equal-tp)}
\]
\[
\text{(?e4 ?INT-AN03-ano25 start-time-point)}
\]
\[
\text{(?e4 in-region-is-office-event) (?e4 time-interval)}
\]
\[
\text{(?e4 ?INT-AN04-ano28 start-time-point)}
\]
\[
\text{(?INT-AN04-ano28 ?INT-AN03-ano29 start-time-point)}
\]
\[
\text{(?e6 ?INT-AN03-ano29 start-time-point)}
\]
\[
\text{(?e6 in-region-is-home-event) (?e6 time-interval)}
\]
\[
\text{(?e1 ?INT-AN02-ano26 end-time-point)}
\]
\[
\text{(?e2 ?INT-AN01-ano27 start-time-point)}
\]
\[
\text{(?INT-AN02-ano26 ?INT-AN01-ano27 tp-before-or-equal-tp)}
\]
\[
\text{(?e1 ?INT-AN02-ano26 START-STATE) (top ?INT-AN02-ano16)}
\]
\[
\text{(?top ?INT-AN02-ano17) (top ?e5) (top ?INT-AN02-ano18)}
\]
\[
\text{(?top ?INT-AN02-ano19) (top ?e1) (top ?INT-AN02-ano20)}
\]
\[
\text{(?top ?INT-AN02-ano21)}
\]
\[
\text{neg}
\]
\[
\text{(:project-to (?e1 ?e2)}
\]
\[
\text{(and (?e-an01-ano30 special-work-day-event)}
\]
\[
\text{(?e-an01-ano30 ?e1 START-STATE)}
\]
\[
\text{(?e-an01-ano30 ?e2 END-STATE))}
\]
\[
\text{(instance (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) special-work-day-event)}
\]
\[
\text{(related (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) ?1 START-STATE)}
\]
\[
\text{(related (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) ?2 END-STATE)}
\]
\[
\text{(related (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) ?1 subevent)}
\]
\[
\text{(related (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) ?2 subevent)}
\]
\[
\text{(related (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) ?e4 subevent)}
\]
\[
\text{(related (:new-ind swd ?e1 ?e2 ?e3 ?e4 ?e5) ?e5 subevent))}
\]
\]
\text{(:box default))}
Conclusion

- Demanding application scenario
- Our approach relies on
  - A mixture of procedural and logical techniques
  - Non-monotonic and non-safe rule language with first-order properties
    - syntactic sugar for universal quantifiers in SPARQL?
  - Tight coupling between rule engine and reasoner
    - rel. DB-based approaches probably don't work here
      (too dynamic and too many DB updates)
  - Generalized Allen relations for definitions
    (e.g. g-future = \{ meets, after \} rather than meets)
    → disjunctions cause blowup in expanded queries (DNF)