What Happened to Bob?

Semantic Data Mining of Context Histories

Michael Wessel, Marko Luther, Ralf Möller

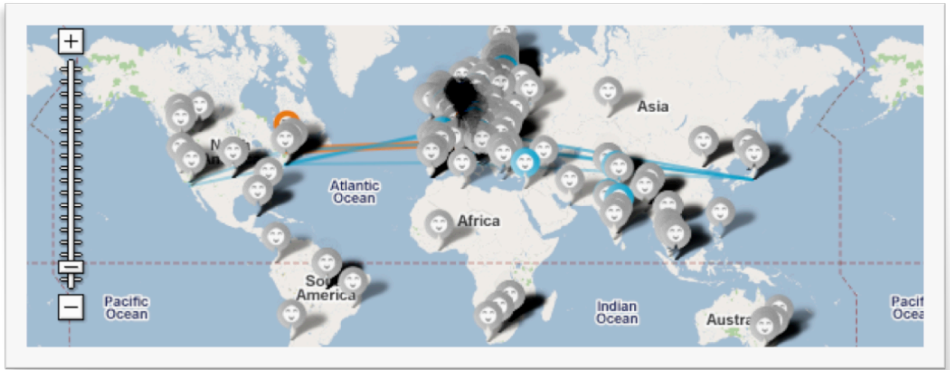
Racer Systems • DOCOMO Euro-Labs • Uni Hamburg

DOCOMO Euro-Labs

www.iyouit.eu

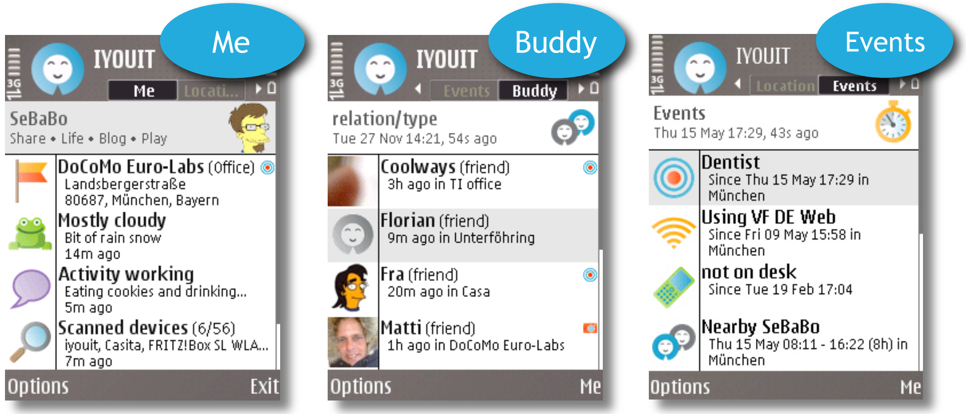
iyouit

share life blog play



A mobile community service

- 1300+ users in 60+ countries
- Connected to emerging Web 2.0 services



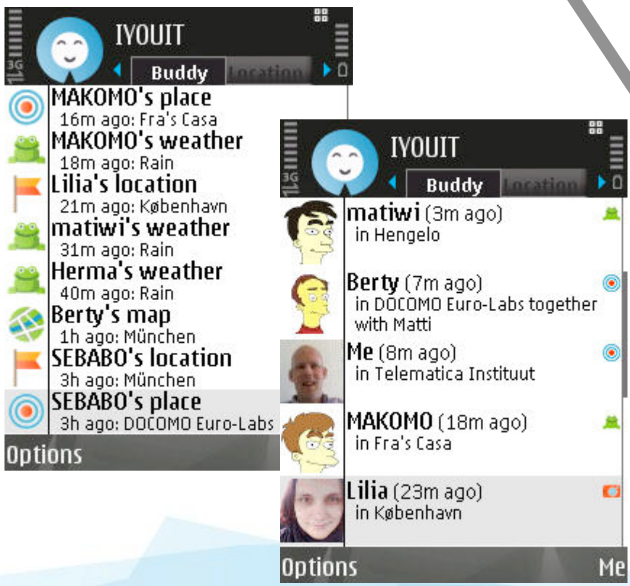
Your digital life recorder

- Facilitating context awareness on standard phones
- Integration of key Semantic Web technologies

flickr · Google Earth · YouTube · last.fm · facebook · twitter · DOPPLR

What for?

Real time context sharing



Life blogging and widgets

Automatic photo & video tagging

ContextWatcher Blogs in 2005

FRIDAY, NOVEMBER 11, 2005

A busy ISWC day

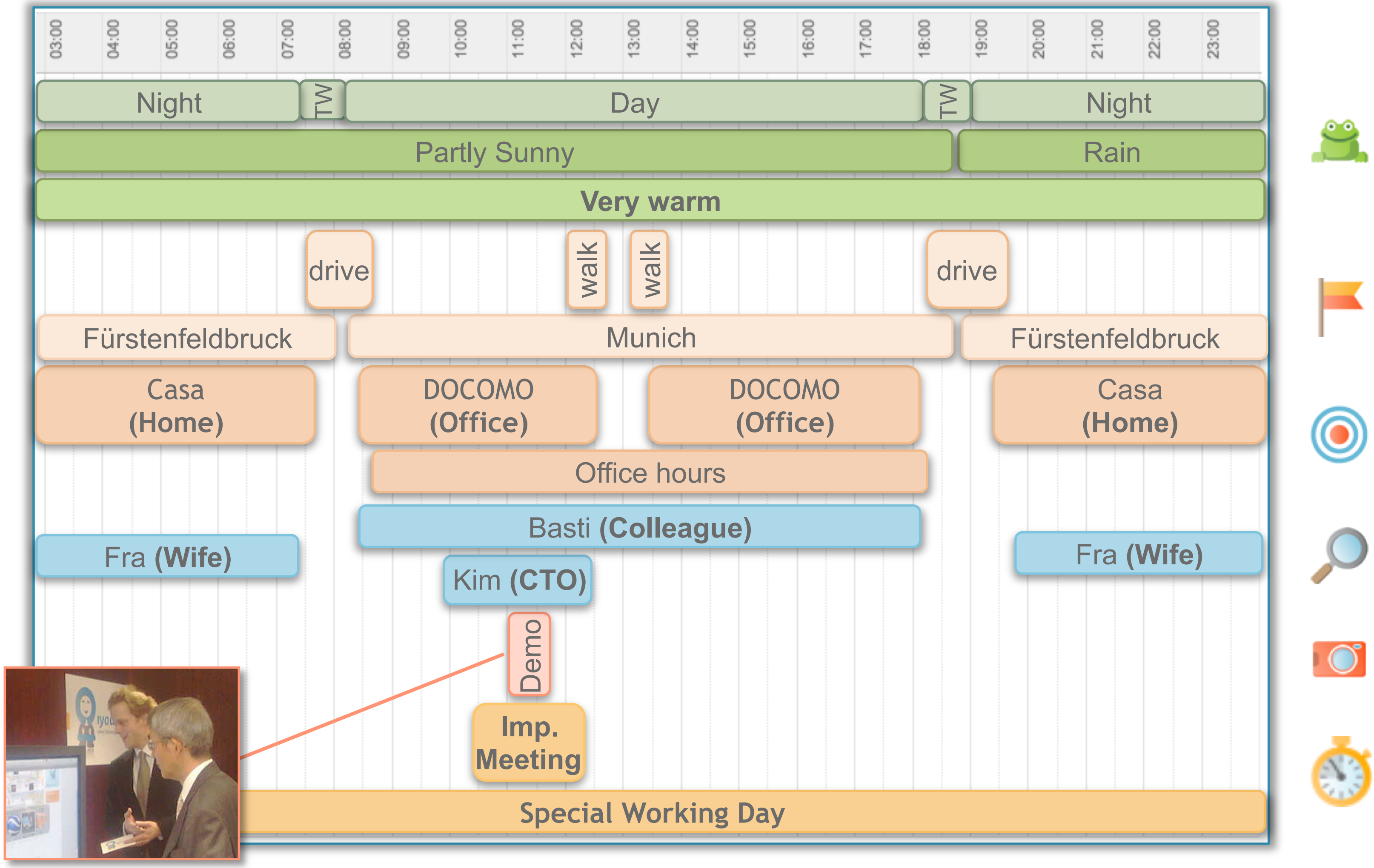
Today was a busy ISWC day (84.1% covered). I took 9 pictures in Dublin and Galway.

I visited Galway (51.8%), München (7.3%), Dublin (18.9%) and Offaly (9.9%), mainly Commute (5.2%) and ISWC (35.1%). I met luther (38.7%). My maximum speed was 131.2 km/h.

POSTED BY MATTHIAS AT 1:01 PM 0 COMMENTS

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.

Towards a Life Browser



Situations & Situation Recognition

- Situations are vectors of attribute-value pairs (CA x CV) in the ABox

$$\{ \; sit : situation, \; (sit, val_1) : CA_1, \; val_1 : CV_1, \ldots, \; (sit, val_n) : CA_n, \; val_n : CV_n \; \}$$

  - + assertions for nearby persons, locations, social networks, …
  - reasoning required to recognize CA x CV occurrences

- Recognition with defined concepts and queries (rules)

$$business\_meeting \doteq \; \geq_3 \; near\_by.colleague \sqcap \exists at\_place.office$$

$$\begin{aligned} ans(x) \leftarrow \; & business\_meeting(x), \\ & near\_by(x, y), near\_by(y, z), near\_by(z, x), \\ & hates(x, y), hates(y, z), hates(z, x). \end{aligned}$$

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

$$simple\_event \doteq \exists start\_state.state \sqcap \exists end\_state.state$$

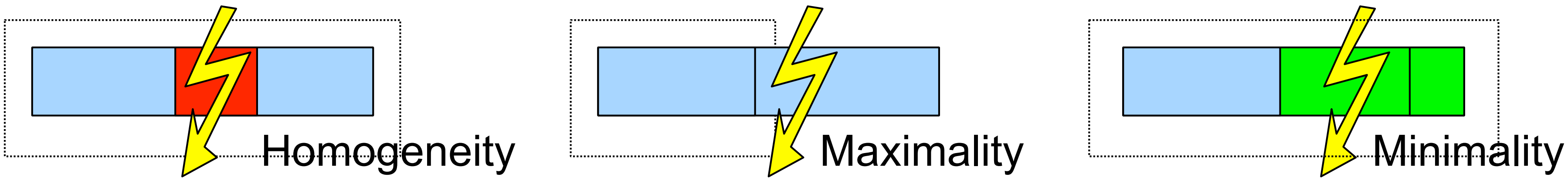
$$complex\_event \doteq event \sqcap \exists has\_part.event$$

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

$$in\_office(p_1), meeting\_with\_boss(p_2), meeting\_with\_customer(p_3),$$
$$meets(p_2, p_3), during(p_2, p_1), during(p_3, p_1)$$

Realization of Event *Recognizers*

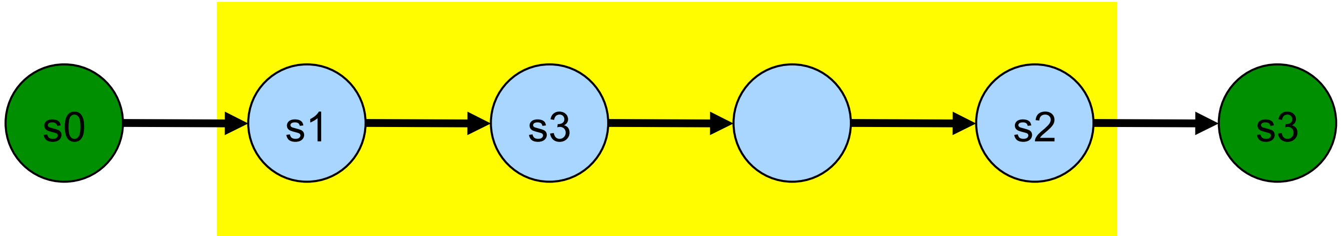
- Assuming events are already present as aggregates in the ABox, how to recognize them?
  - Defined concepts
    - Problems with relational expressivity for complex events
      - only tree-shaped temporal constraints expressible
      - Important event properties cannot be expressed (required for definitions of complex events!)



  - Queries and rules
    - High relational expressivity over the ABox
    - Universal closed-domain quantifier (SPARQL, SQL, nRQL)

Recognizing Homogeneous Events (1)

$$
\begin{aligned}
ans(s_1, s_2) \leftarrow \quad & state(s_1), state(s_2), future(s_1, s_2), \\
& P(s_1), P(s_2), \\
& \backslash\pi(s_1)\ (state(s_0), next(s_0, s_1), P(s_0)), \\
& \backslash\pi(s_2)\ (state(s_3), next(s_2, s_3), P(s_3)), \\
& \backslash\pi(s_1, s_2)\ (\quad state(s_3), future(s_1, s_3), future(s_3, s_2), \\
& \qquad\qquad\qquad \backslash P(s_3))
\end{aligned}
$$

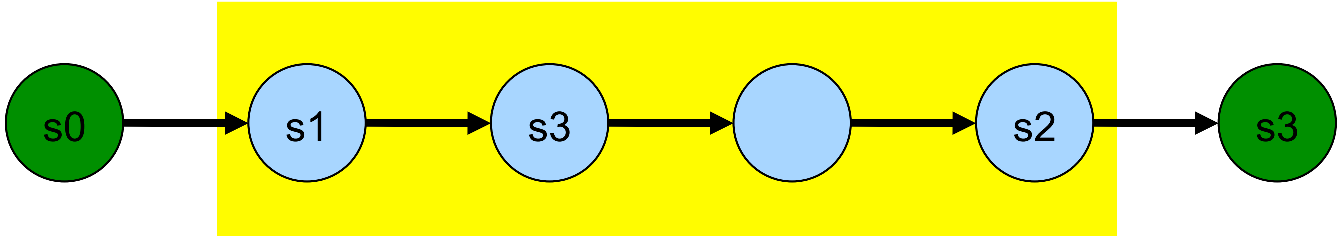



```
(retrieve (?s1 ?s2)
  (and (?s1 state) (?s2 state) (?s1 ?s2 future)
          (?s1 P) (?s2 P)
          (neg (project-to (?s1) (and (?s0 state) (?s0 ?s1 next) (?so P)))
          (neg (project-to (?s2) (and (?s3 state) (?s2 ?s3 next) (?s3 P)))
          (neg (project-to (?s1 ?s2) (and (?s3 state)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

Recognizing Homogeneous Events (2)

$$\mathcal{S}_{\mathcal{A}} = (\Delta^{\mathcal{I}}, C^{\mathcal{I}}, ..., R^{\mathcal{I}}, ...), \text{ with}$$

$$\begin{aligned}
\Delta^{\mathcal{I}} &= \mathsf{inds}(\mathcal{A}), \\
C^{\mathcal{I}} &= \{\, i \mid \; i \in \mathsf{inds}(\mathcal{A}), \mathcal{A} \models C(i) \,\} \;\;, \\
R^{\mathcal{I}} &= \{\, (i,j) \mid \; \mathsf{inds}(\mathcal{A}), \mathcal{A} \models R(i,j) \,\},
\end{aligned}$$



$$\begin{aligned}
\{\, (s_1, s_2) \mid \; \exists s_1, s_2 : \quad & \mathit{state}(s_1) \wedge \mathit{state}(s_1) \wedge \mathit{future}(s_1, s_2) \;\wedge \\
& P(s_1) \wedge P(s_2) \;\wedge \\
& \neg \exists s_0 : \quad \mathit{state}(s_0) \wedge \mathit{next}(s_0, s_1) \;\; \wedge P(s_0) \;\wedge \\
& \neg \exists s_3 : \quad \mathit{state}(s_3) \wedge \mathit{next}(s_2, s_3) \;\; \wedge P(s_3) \;\wedge \\
& \neg \exists s_3 : \quad \mathit{state}(s_3) \wedge \mathit{future}(s_1, s_3), \mathit{future}(s_3, s_2) \;\wedge \\
& \qquad\quad \neg P(s_3) \,\}
\end{aligned}$$

Where do Events Come From?

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

$$at\_home \sqcap \exists future.in\_office \dot{\sqsubseteq} \exists part\_of.(home\_2\_office\_event \sqcap \exists has\_part.in\_office)$$

- 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

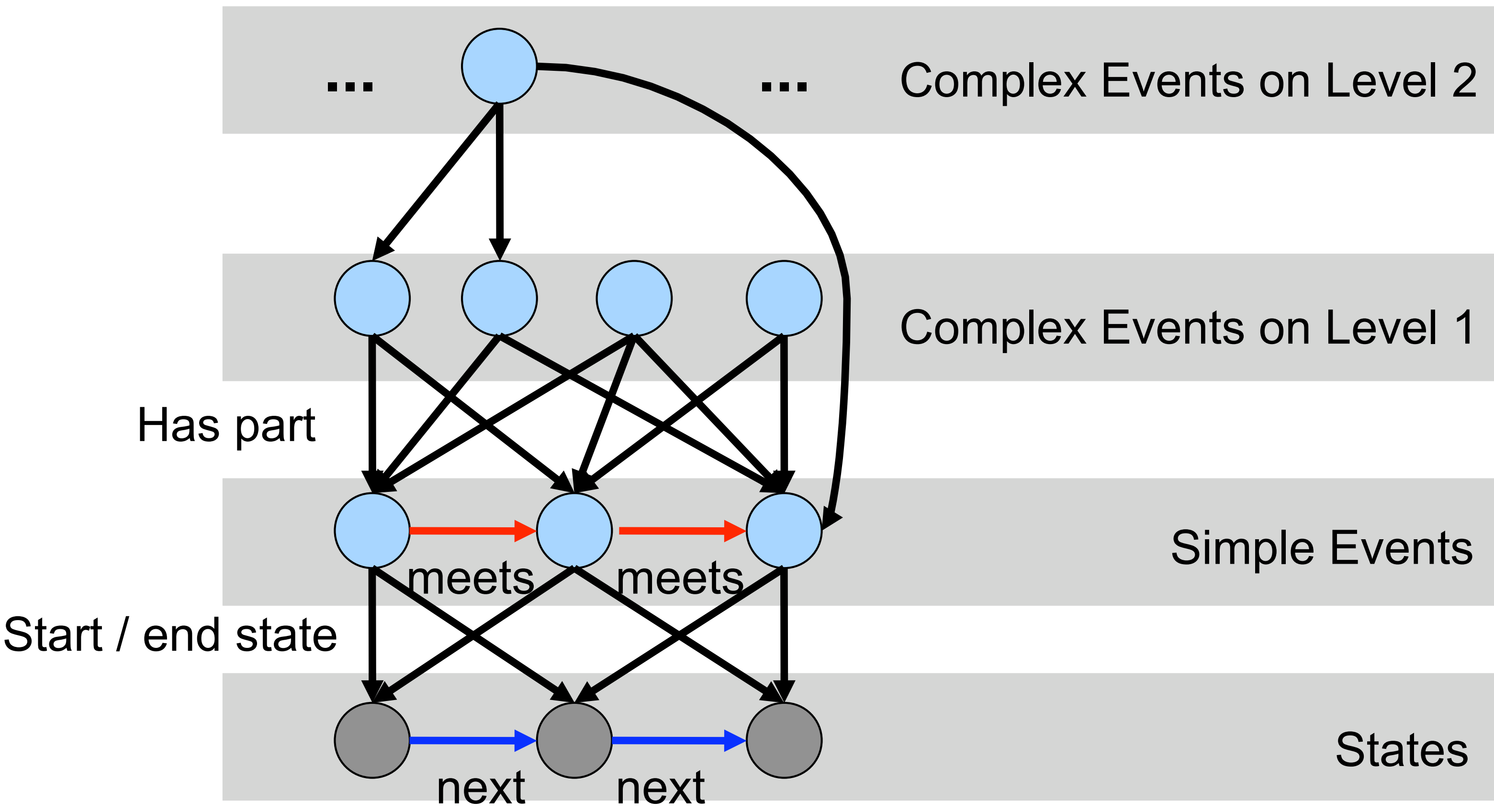
Approach 1- Pre-constructed Events

- 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**

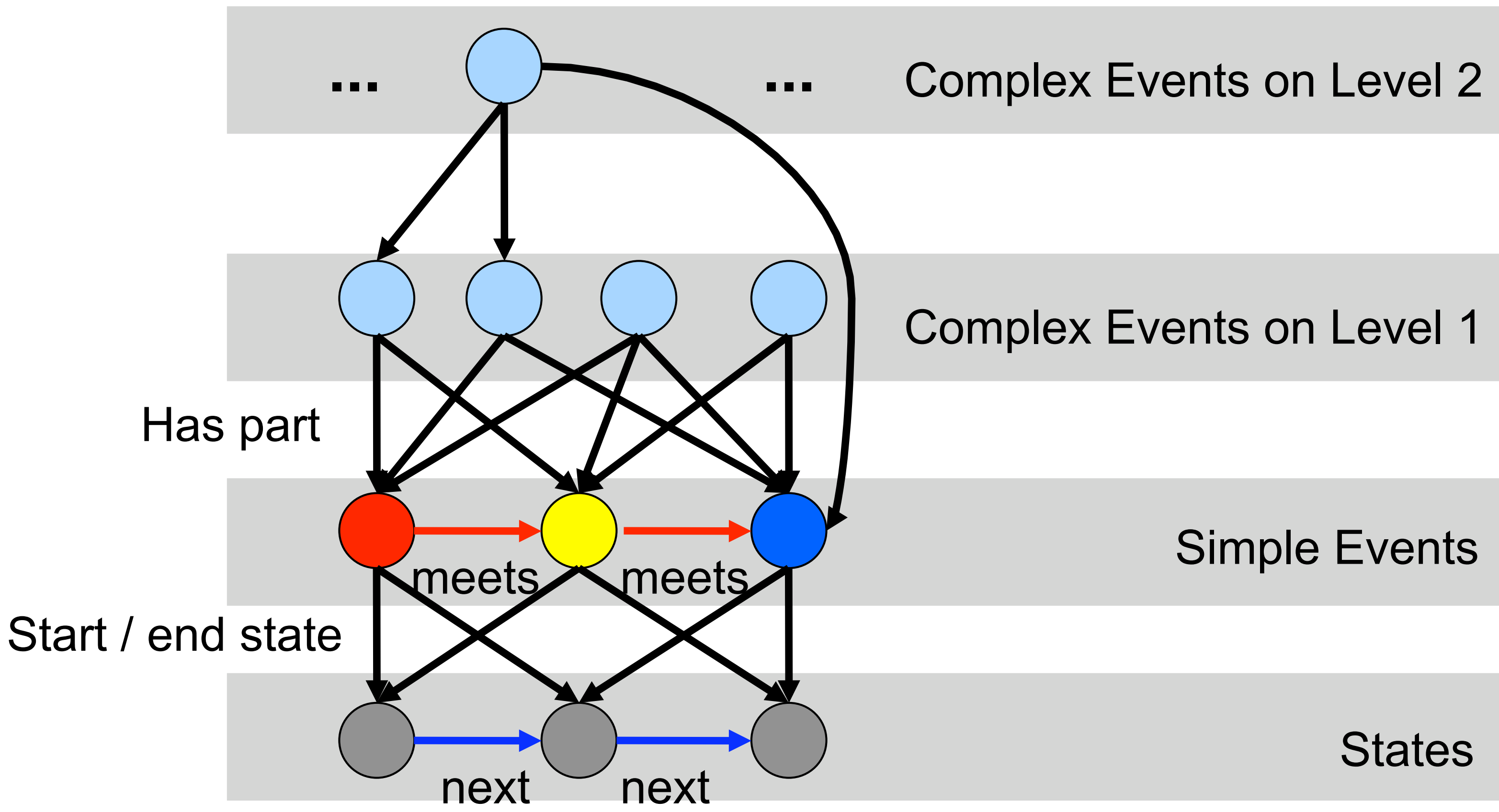
$$\begin{aligned} \mathit{stressful\_office\_day}(x) \leftarrow \quad & \mathit{has\_part}(x, p_1), \mathit{has\_part}(x, p_2), \mathit{has\_part}(x, p_3), \\ & \mathit{in\_office}(p_1), \mathit{meeting\_with\_boss}(p_2), \\ & \mathit{meeting\_with\_customer}(p_3), \\ & \mathit{meets}(p_2, p_3), \mathit{during}(p_2, p_1), \mathit{during}(p_3, p_1) \end{aligned}$$

  - Sometimes, defined concepts are sufficient

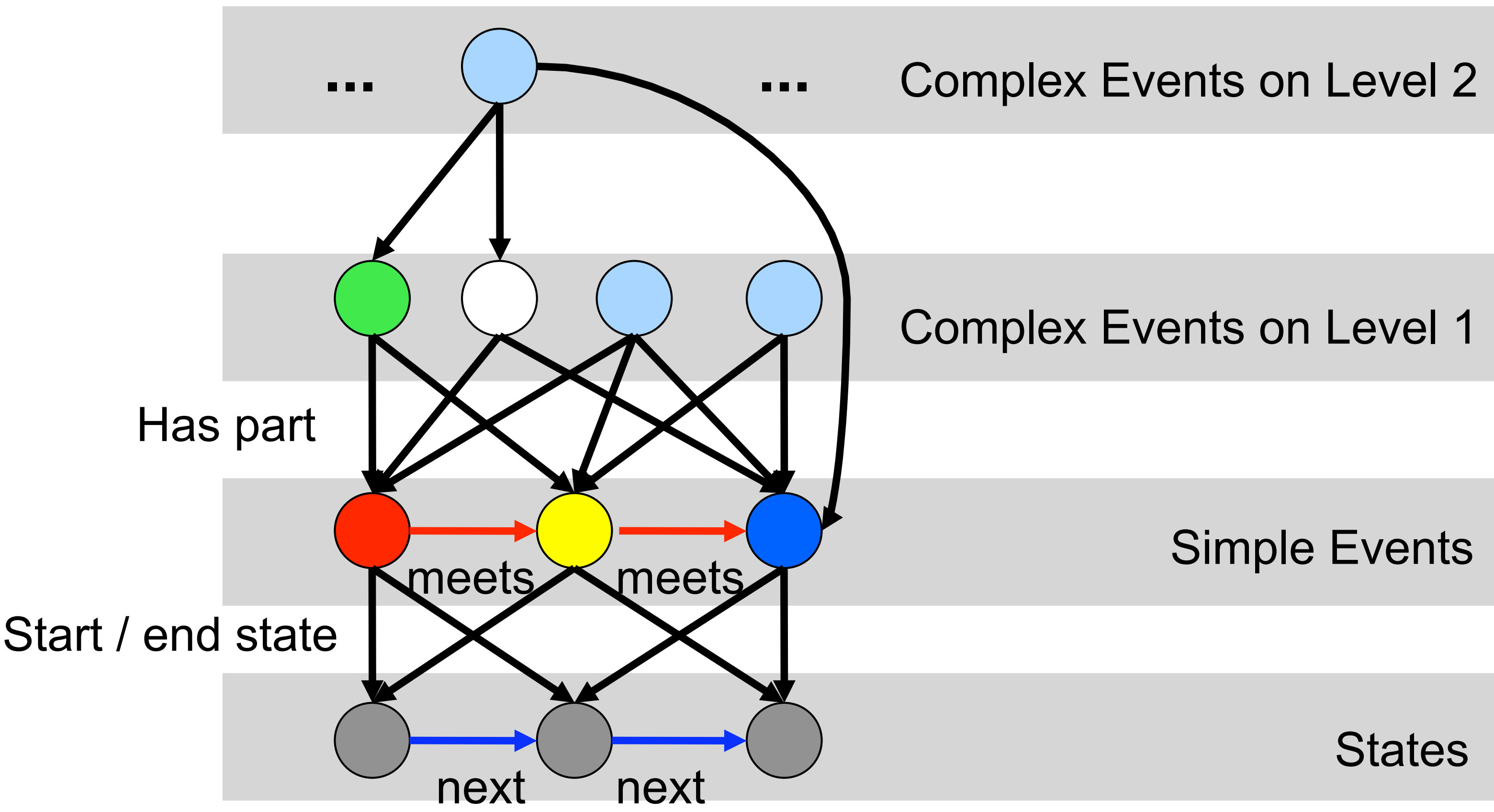
Approach 1- Pre-constructed Events



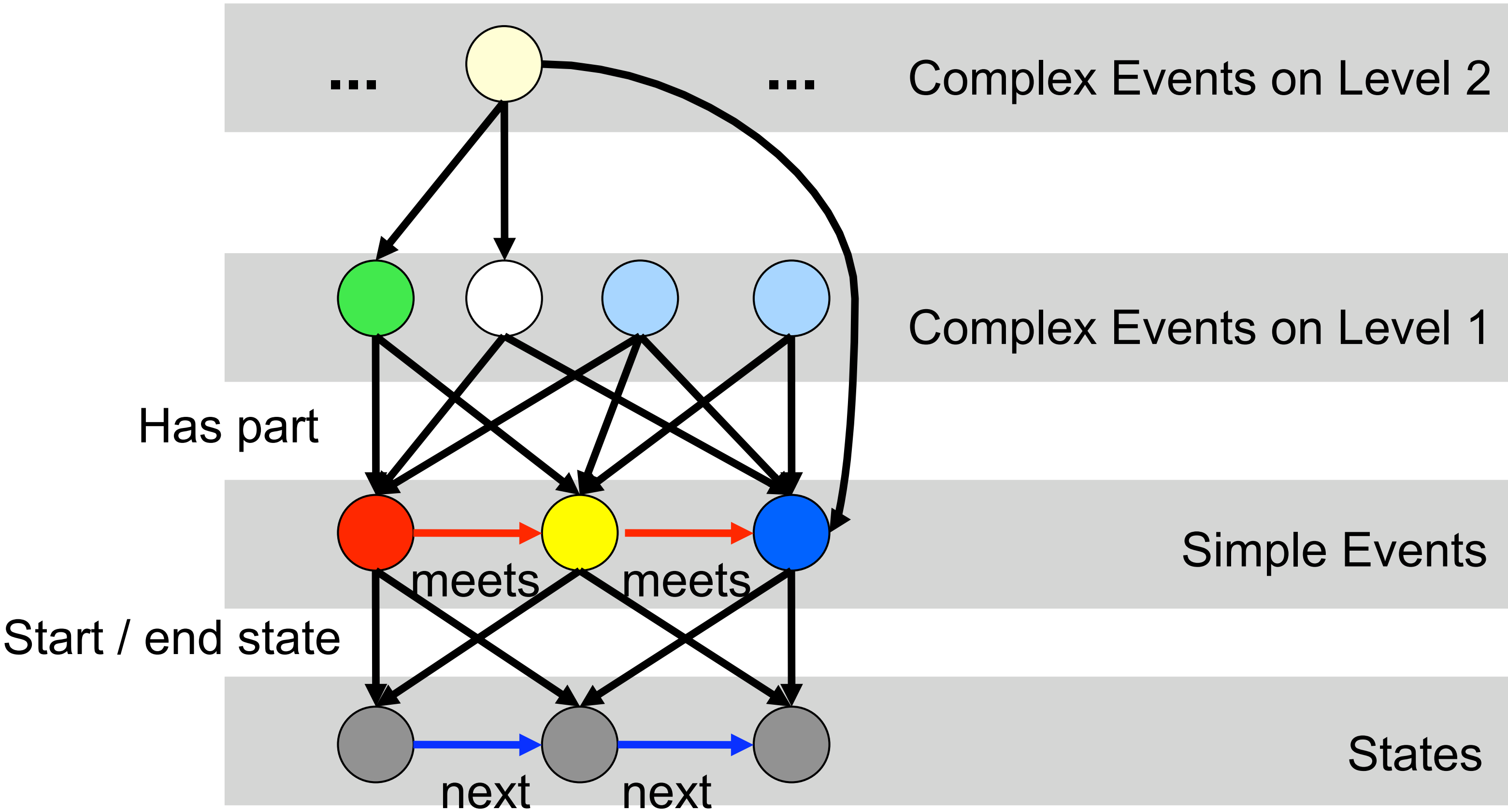
Approach 1- Pre-constructed Events



Approach 1- Pre-constructed Events



Approach 1- Pre-constructed Events



→ Not feasible due to the enormous ABox sizes (too many irrelevant events)

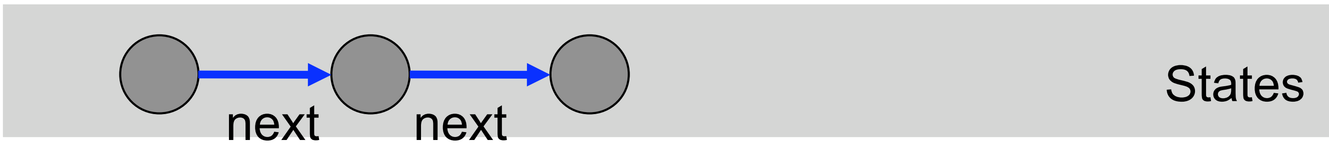
Approach 2 – Constructing Events

- All events are constructed dynamically
  - **Non-safe & non-horn rules** required, e.g. nRQL rules

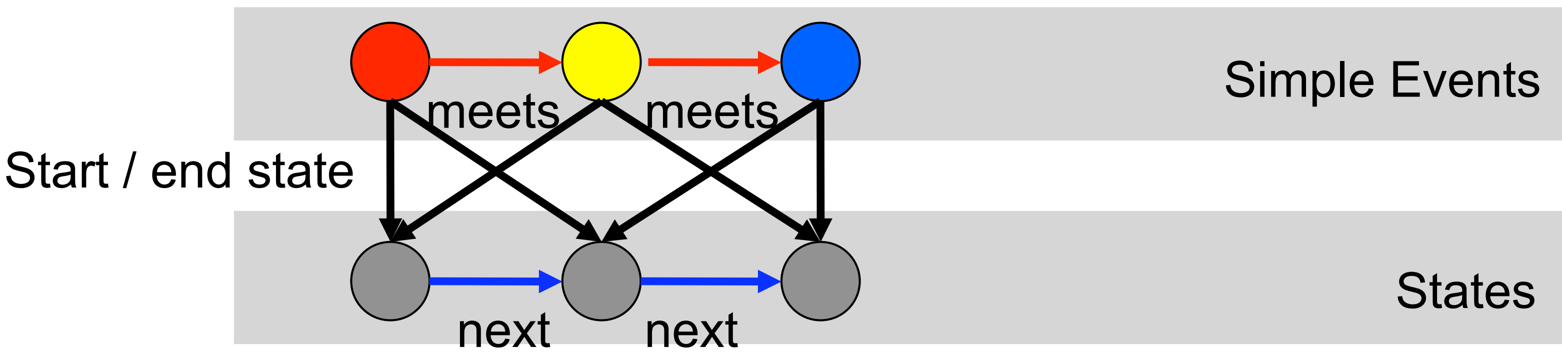
$$
\begin{aligned}
& stressful\_office\_day(new(sod, p_1, p_2, p_3)), \\
& has\_part(new(sod, p_1, p_2, p_3), p_1), \\
& has\_part(new(sod, p_1, p_2, p_3), p_2), \\
& has\_part(new(sod, p_1, p_2, p_3), p_3) \\
& \quad \leftarrow \quad in\_office(p_1), meeting\_with\_boss(p_2), \\
& \qquad\qquad meeting\_with\_customer(p_3), \\
& \qquad\qquad meets(p_2, p_3), during(p_2, p_1), during(p_3, p_1)
\end{aligned}
$$

  - Termination: make rules non-monotonic (acyclic definitions)
    - Construct only one $stressful\_office\_day$ per p_1, p_2, p_3
    - add NAF-negated conclusion to precondition

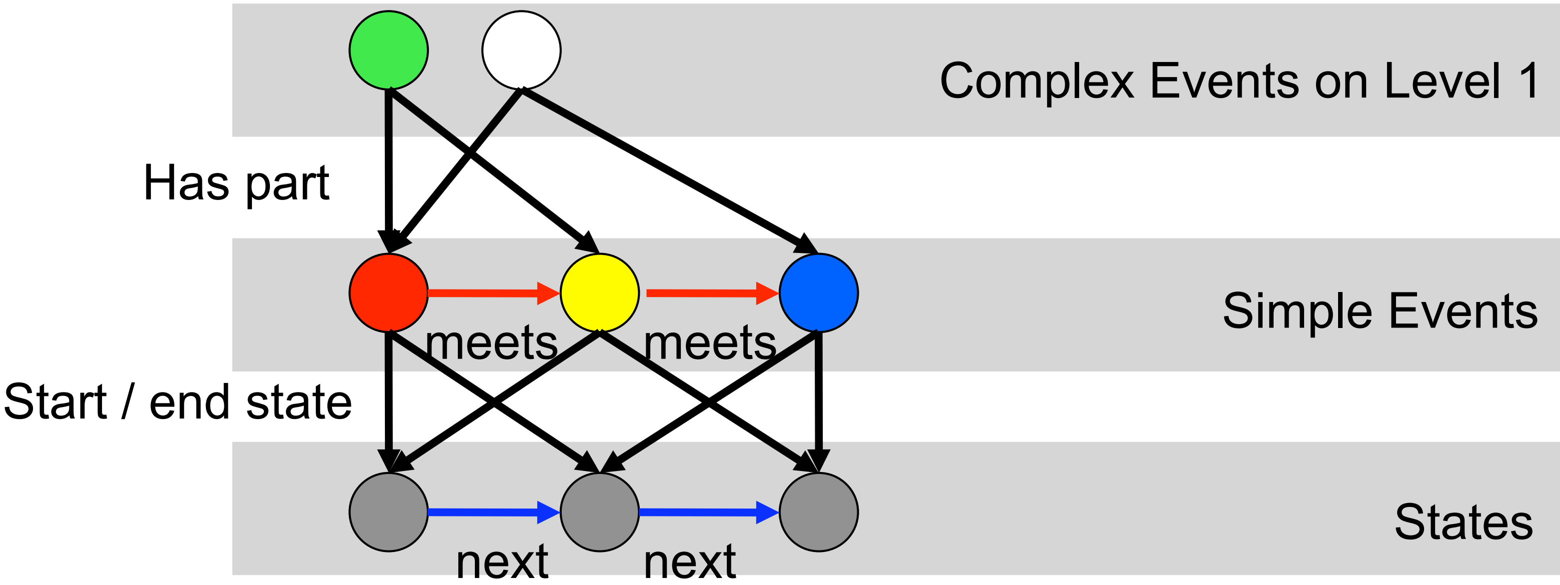
Approach 2 – Constructing Events



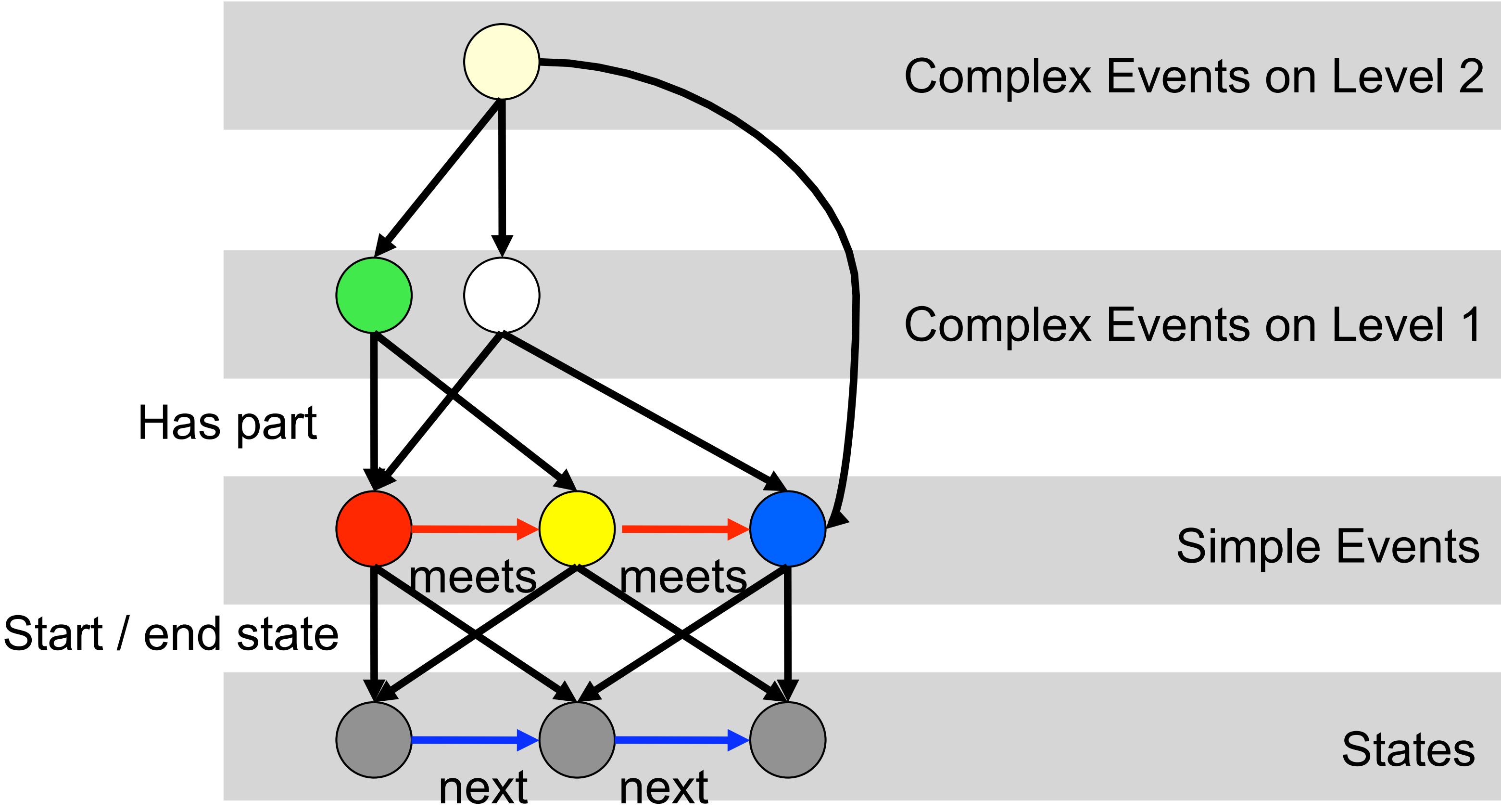
Approach 2 – Constructing Events



Approach 2 – Constructing Events



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

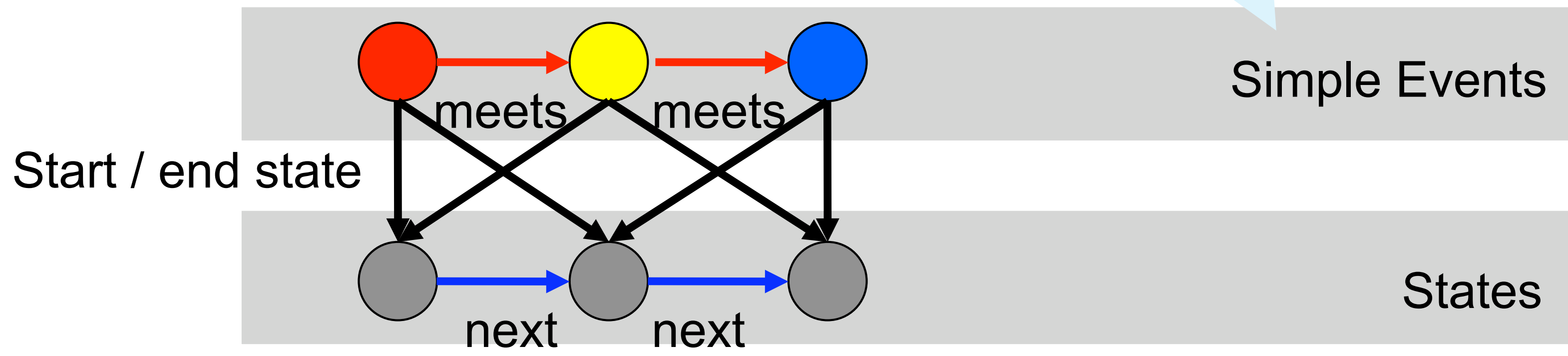
→ Keeps number of required rules small

  - 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)

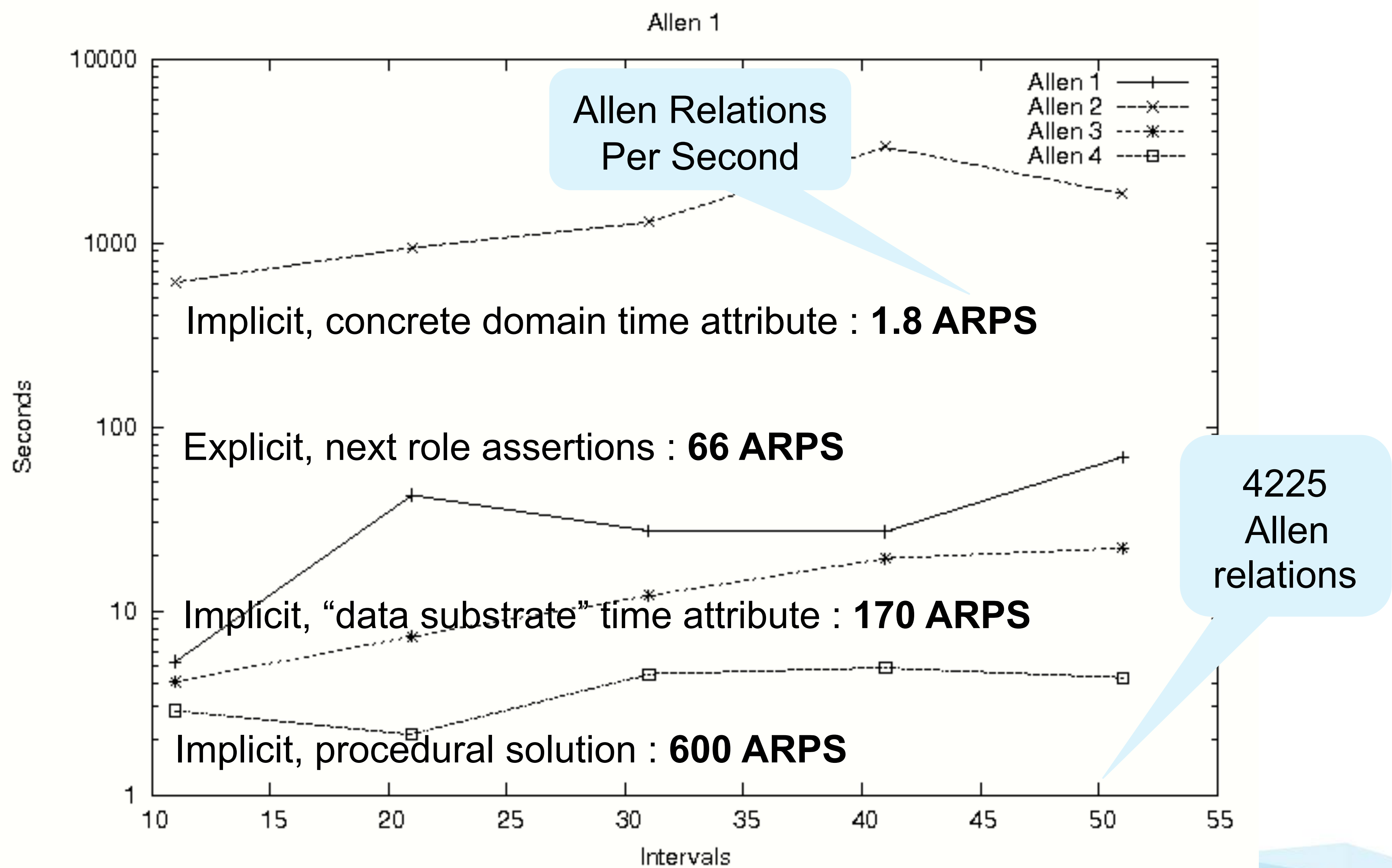
Simple events whose recognition requires reasoning or complex events are constructed dynamically



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



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)

```
(union
   (and (?e1 in-region-is-home-event) (?e1 time-interval)
        (?e3 business-lunch-event) (?e3 time-interval)
        (?e3 ?INT-ANO1-ano23 start-time-point)
        (?INT-ANO2-ano22 ?INT-ANO1-ano23 tp-before-or-equal-tp)
        (?e2 ?INT-ANO2-ano22 end-time-point)
        (?e2 in-region-is-office-event) (?e2 time-interval)
        (?e3 ?INT-ANO4-ano24 end-time-point)
        (?INT-ANO4-ano24 ?INT-ANO3-ano25 tp-before-or-equal-tp)
        (?e4 ?INT-ANO3-ano25 start-time-point)
        (?e4 in-region-is-office-event) (?e4 time-interval)
        (?e4 ?INT-ANO14-ano28 end-time-point)
        (?INT-ANO14-ano28 ?INT-ANO13-ano29 tp-before-or-equal-tp)
        (?e5 ?INT-ANO13-ano29 start-time-point)
        (?e5 in-region-is-home-event) (?e5 time-interval)
        (?e1 ?INT-ANO12-ano26 end-time-point)
        (?e2 ?INT-ANO11-ano27 start-time-point)
        (?INT-ANO12-ano26 ?INT-ANO11-ano27 tp-before-or-equal-tp)
        (?e1 ?s1 START-STATE) (?e5 ?s2 END-STATE) (top ?INT-ANO9-ano16)
        (top ?INT-ANO10-ano17) (top ?e5) (top ?INT-ANO7-ano18)
        (top ?INT-ANO8-ano19) (top ?e1) (top ?INT-ANO5-ano20)
        (top ?INT-ANO6-ano21)
        (neg
         (:project-to (?s1 ?s2)
          (and (?e-ano15-ano30 special-work-day-event)
               (?e-ano15-ano30 ?s1 START-STATE)
               (?e-ano15-ano30 ?s2 END-STATE)))))
   (and (?e1 in-region-is-home-event) (?e1 time-interval)
        (?e3 nearby-is-supervisor-event) (?e3 time-interval)
        (?e3 ?INT-ANO10-ano17 end-time-point)
        (?INT-ANO10-ano17 ?INT-ANO9-ano16 tp-before-or-equal-tp)
        (?e5 ?INT-ANO9-ano16 start-time-point)
        (?e5 in-region-is-home-event) (?e5 time-interval)
        (?e1 ?INT-ANO12-ano26 end-time-point)
        (?INT-ANO12-ano26 ?INT-ANO11-ano27 tp-before-or-equal-tp)
        (?e2 ?INT-ANO11-ano27 start-time-point)
        (?e2 in-region-is-office-event) (?e2 time-interval)
        (?e2 ?INT-ANO6-ano21 end-time-point)
        (?INT-ANO6-ano21 ?INT-ANO5-ano20 tp-before-or-equal-tp)
        (?e4 ?INT-ANO5-ano20 start-time-point)
        (?e4 in-region-is-office-event) (?e4 time-interval)
        (?e1 ?INT-ANO8-ano19 end-time-point)
        (?e3 ?INT-ANO7-ano18 start-time-point)
        (?INT-ANO8-ano19 ?INT-ANO7-ano18 tp-before-or-equal-tp)
        (?e4 ?INT-ANO14-ano28 end-time-point)
        (?e5 ?INT-ANO13-ano29 start-time-point)
        (?INT-ANO14-ano28 ?INT-ANO13-ano29 tp-before-or-equal-tp)
        (?e1 ?s1 START-STATE) (?e5 ?s2 END-STATE) (top ?INT-ANO3-ano25)
        (top ?INT-ANO4-ano24) (top ?INT-ANO1-ano23)
        (top ?INT-ANO2-ano22)
        (neg
         (:project-to (?s1 ?s2)
          (and (?e-ano15-ano30 special-work-day-event)
               (?e-ano15-ano30 ?s1 START-STATE)
               (?e-ano15-ano30 ?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))
```

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)