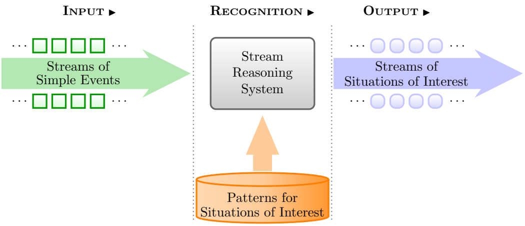
Temporal Specification Optimisation for the Event Calculus

Periklis Mantenoglou   Alexander Artikis

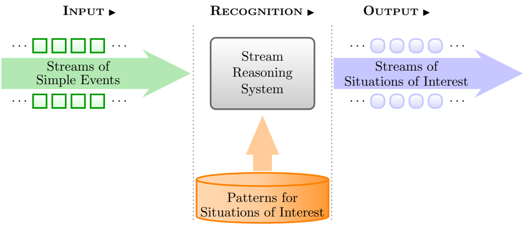
NCSR Demokritos, Greece

http://cer.iit.demokritos.gr/

Temporal Pattern Matching over Streams



Temporal Pattern Matching over Streams



https://cer.iit.demokritos.gr (activity recognition)

Event Calculus

- A logic programming language for representing and reasoning about events and their effects.
- Key components:
  - event (typically instantaneous).
  - fluent: a property that may have different values at different points in time.

Robert A. Kowalski, Marek J. Sergot: A Logic-based Calculus of Events. New Gener. Comput. 4(1): 67-95, 1986.

Event Calculus

- A logic programming language for representing and reasoning about events and their effects.
- Key components:
  - event (typically instantaneous).
  - fluent: a property that may have different values at different points in time.
- Built-in representation of inertia:
  - $F = V$ holds at a particular time-point if $F = V$ has been *initiated* by an event at some earlier time-point, and not *terminated* by another event in the meantime.

Robert A. Kowalski, Marek J. Sergot: A Logic-based Calculus of Events. New Gener. Comput. 4(1): 67-95, 1986.

Run-Time Event Calculus (RTEC)

Simple Fluent (SF):

initiatedAt$(F{=}V, T) \leftarrow$
    happensAt(E_{In_1}, T)[,
    conditions].
    ⋮

terminatedAt$(F{=}V, T) \leftarrow$
    happensAt(E_{T_1}, T)[,
    conditions].
    ⋮

where conditions:
$^{0-K}$[not] **happensAt**(E_k, T),
$^{0-M}$[not] **holdsAt**$(F_m = V_m, T)$,
$^{0-N}$atemporal-constraint$_n$

Artikis A., Sergot M. and Paliouras G., An Event Calculus for Event Recognition. In IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(4), 895–908, 2015.

Run-Time Event Calculus (RTEC)

Simple Fluent (SF):

$\textbf{initiatedAt}(F{=}V, T) \leftarrow$
$\quad \textbf{happensAt}(E_{In_1}, T)[,$
$\quad \text{conditions}].$

$\vdots$

$\textbf{terminatedAt}(F{=}V, T) \leftarrow$
$\quad \textbf{happensAt}(E_{T_1}, T)[,$
$\quad \text{conditions}].$

$\vdots$

where conditions:

$^{0-K}[\text{not}]\ \textbf{happensAt}(E_k, T),$
$^{0-M}[\text{not}]\ \textbf{holdsAt}(F_m = V_m, T),$
$^{0-N}\text{atemporal-constraint}_n$

Statically Determined Fluent (SDF):

$\textbf{holdsFor}(F{=}V, I) \leftarrow$
$\quad \textbf{holdsFor}(F_1 = V_1, I_1)[,$
$\quad \textbf{holdsFor}(F_2 = V_2, I_2), \ldots$
$\quad \textbf{holdsFor}(F_n = V_n, I_n),$
$\quad \text{intervalConstruct}(L_1, I_{n+1}), \ldots$
$\quad \text{intervalConstruct}(L_m, I)].$

where intervalConstruct:

union_all or
intersect_all or
relative_complement_all

Artikis A., Sergot M. and Paliouras G., An Event Calculus for Event Recognition. In IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(4), 895–908, 2015.

Run-Time Event Calculus (RTEC)

Simple Fluent (SF):

$\textbf{initiatedAt}(F = V, T) \leftarrow$
$\quad \textbf{happensAt}(E_{In_1}, T)[,$
$\quad \text{conditions}].$
$\quad \vdots$
$\textbf{terminatedAt}(F = V, T) \leftarrow$
$\quad \textbf{happensAt}(E_{T_1}, T)[,$
$\quad \text{conditions}].$
$\quad \vdots$

where conditions:
${}^{0-K}[\text{not}]\ \textbf{happensAt}(E_k, T),$
${}^{0-M}[\text{not}]\ \textbf{holdsAt}(F_m = V_m, T),$
${}^{0-N}\text{atemporal-constraint}_n$

- SFs $\supseteq$ SDFs.

Statically Determined Fluent (SDF):

$\textbf{holdsFor}(F = V, I) \leftarrow$
$\quad \textbf{holdsFor}(F_1 = V_1, I_1)[,$
$\quad \textbf{holdsFor}(F_2 = V_2, I_2), \ldots$
$\quad \textbf{holdsFor}(F_n = V_n, I_n),$
$\quad \text{intervalConstruct}(L_1, I_{n+1}), \ldots$
$\quad \text{intervalConstruct}(L_m, I)].$

where intervalConstruct:
union_all or
intersect_all or
relative_complement_all

Artikis A., Sergot M. and Paliouras G., An Event Calculus for Event Recognition. In IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(4), 895–908, 2015.

Run-Time Event Calculus (RTEC)

Simple Fluent (SF):

initiatedAt$(F{=}V, T) \leftarrow$
  happensAt(E_{In_1}, T)[,
  conditions].
  ⋮

terminatedAt$(F{=}V, T) \leftarrow$
  happensAt(E_{T_1}, T)[,
  conditions].
  ⋮

where conditions:
$^{0-K}$[not] **happensAt**(E_k, T),
$^{0-M}$[not] **holdsAt**$(F_m = V_m, T)$,
$^{0-N}$atemporal-constraint$_n$

- SFs $\supseteq$ SDFs.

Statically Determined Fluent (SDF):

holdsFor$(F{=}V, I) \leftarrow$
  holdsFor$(F_1 = V_1, I_1)$[,
  holdsFor$(F_2 = V_2, I_2), \ldots$
  holdsFor$(F_n = V_n, I_n)$,
  intervalConstruct$(L_1, I_{n+1}), \ldots$
  intervalConstruct(L_m, I)].

where intervalConstruct:
  union_all or
  intersect_all or
  relative_complement_all

- SDFs:
  - exponentially more compact.
  - more efficient to reason with.

Artikis A., Sergot M. and Paliouras G., An Event Calculus for Event Recognition. In IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(4), 895–908, 2015.

Problem Statement & Proposed Solution

Challenges:

- Most Event Calculus specifications contain only SFs.

Problem Statement & Proposed Solution

Challenges:

- Most Event Calculus specifications contain only SFs.
- The knowledge engineer may detect only a portion of SFs that can be re-written as equivalent SDFs.

Problem Statement & Proposed Solution

Challenges:

- Most Event Calculus specifications contain only SFs.
- The knowledge engineer may detect only a portion of SFs that can be re-written as equivalent SDFs.

Our Approach:

- Formal characterisation of the class of SFs that are translatable into equivalent SDFs.

Problem Statement & Proposed Solution

Challenges:

- Most Event Calculus specifications contain only SFs.
- The knowledge engineer may detect only a portion of SFs that can be re-written as equivalent SDFs.

Our Approach:

- Formal characterisation of the class of SFs that are translatable into equivalent SDFs.
- Compiler that identifies and re-writes them as SDFs.

Problem Statement & Proposed Solution

Challenges:

- Most Event Calculus specifications contain only SFs.
- The knowledge engineer may detect only a portion of SFs that can be re-written as equivalent SDFs.

Our Approach:

- Formal characterisation of the class of SFs that are translatable into equivalent SDFs.
- Compiler that identifies and re-writes them as SDFs.
- Reproducible empirical evaluation on numerous real domain specifications.

Example: A Translatable SF

SF:

$\textbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\mathit{active}(P_1) = \text{true}), T),$
$\quad \textbf{holdsAt}(\mathit{close}(P_1, P_2) = \text{true}, T),$
$\quad \text{not } \textbf{happensAt}(\text{end}(\mathit{close}(P_1, P_2) = \text{true}), T).$

$\textbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\mathit{close}(P_1, P_2) = \text{true}), T),$
$\quad \textbf{holdsAt}(\mathit{active}(P_1) = \text{true}, T),$
$\quad \text{not } \textbf{happensAt}(\text{end}(\mathit{active}(P_1) = \text{true}), T).$

$\textbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\mathit{active}(P_1) = \text{true}), T),$
$\quad \textbf{happensAt}(\text{start}(\mathit{close}(P_1, P_2) = \text{true}), T).$

$\textbf{terminatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{end}(\mathit{active}(P_1) = \text{true}), T).$

$\textbf{terminatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{end}(\mathit{close}(P_1, P_2) = \text{true}), T).$

Example: A Translatable SF

SF:

$\textbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\mathit{active}(P_1) = \text{true}), T),$
$\quad \textbf{holdsAt}(\mathit{close}(P_1, P_2) = \text{true}, T),$
$\quad \text{not } \textbf{happensAt}(\text{end}(\mathit{close}(P_1, P_2) = \text{true}), T).$

$\textbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\mathit{close}(P_1, P_2) = \text{true}), T),$
$\quad \textbf{holdsAt}(\mathit{active}(P_1) = \text{true}, T),$
$\quad \text{not } \textbf{happensAt}(\text{end}(\mathit{active}(P_1) = \text{true}), T).$

$\textbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\mathit{active}(P_1) = \text{true}), T),$
$\quad \textbf{happensAt}(\text{start}(\mathit{close}(P_1, P_2) = \text{true}), T).$

$\textbf{terminatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{end}(\mathit{active}(P_1) = \text{true}), T).$

$\textbf{terminatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{end}(\mathit{close}(P_1, P_2) = \text{true}), T).$

SDF:

$\textbf{holdsFor}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, I) \leftarrow$
$\quad \textbf{holdsFor}(\mathit{active}(P_1) = \text{true}, I_a),$
$\quad \textbf{holdsFor}(\mathit{close}(P_1, P_2) = \text{true}, I_c),$
$\quad \textbf{intersect\_all}([I_a, I_c], I).$

Example: A Translatable SF

SF:

$\textbf{initiatedAt}(\textit{meeting}(P_1, P_2) = \textit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\textit{active}(P_1) = \text{true}), T),$
$\quad \textbf{holdsAt}(\textit{close}(P_1, P_2) = \text{true}, T),$
$\quad \text{not } \textbf{happensAt}(\text{end}(\textit{close}(P_1, P_2) = \text{true}), T).$

$\textbf{initiatedAt}(\textit{meeting}(P_1, P_2) = \textit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\textit{close}(P_1, P_2) = \text{true}), T),$
$\quad \textbf{holdsAt}(\textit{active}(P_1) = \text{true}, T),$
$\quad \text{not } \textbf{happensAt}(\text{end}(\textit{active}(P_1) = \text{true}), T).$

$\textbf{initiatedAt}(\textit{meeting}(P_1, P_2) = \textit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\textit{active}(P_1) = \text{true}), T),$
$\quad \textbf{happensAt}(\text{start}(\textit{close}(P_1, P_2) = \text{true}), T).$

$\textbf{terminatedAt}(\textit{meeting}(P_1, P_2) = \textit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{end}(\textit{active}(P_1) = \text{true}), T).$

$\textbf{terminatedAt}(\textit{meeting}(P_1, P_2) = \textit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{end}(\textit{close}(P_1, P_2) = \text{true}), T).$

SDF:

$\textbf{holdsFor}(\textit{meeting}(P_1, P_2) = \textit{interact}, I) \leftarrow$
$\quad \textbf{holdsFor}(\textit{active}(P_1) = \text{true}, I_a),$
$\quad \textbf{holdsFor}(\textit{close}(P_1, P_2) = \text{true}, I_c),$
$\quad \textbf{intersect\_all}([I_a, I_c], I).$

SDF is satisfied iff we have: $\textit{active}(P_1) = \text{true} \wedge \textit{close}(P_1, P_2) = \text{true}$.

Example: A Translatable SF

SF:

$\mathbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \mathbf{happensAt}(\mathrm{start}(\mathit{active}(P_1) = \mathrm{true}), T),$
$\quad \mathbf{holdsAt}(\mathit{close}(P_1, P_2) = \mathrm{true}, T),$
$\quad \mathrm{not}\ \mathbf{happensAt}(\mathrm{end}(\mathit{close}(P_1, P_2) = \mathrm{true}), T).$

$\mathbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \mathbf{happensAt}(\mathrm{start}(\mathit{close}(P_1, P_2) = \mathrm{true}), T),$
$\quad \mathbf{holdsAt}(\mathit{active}(P_1) = \mathrm{true}, T),$
$\quad \mathrm{not}\ \mathbf{happensAt}(\mathrm{end}(\mathit{active}(P_1) = \mathrm{true}), T).$

$\mathbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \mathbf{happensAt}(\mathrm{start}(\mathit{active}(P_1) = \mathrm{true}), T),$
$\quad \mathbf{happensAt}(\mathrm{start}(\mathit{close}(P_1, P_2) = \mathrm{true}), T).$

$\mathbf{terminatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \mathbf{happensAt}(\mathrm{end}(\mathit{active}(P_1) = \mathrm{true}), T).$

$\mathbf{terminatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \mathbf{happensAt}(\mathrm{end}(\mathit{close}(P_1, P_2) = \mathrm{true}), T).$

SDF:

$\mathbf{holdsFor}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, I) \leftarrow$
$\quad \mathbf{holdsFor}(\mathit{active}(P_1) = \mathrm{true}, I_a),$
$\quad \mathbf{holdsFor}(\mathit{close}(P_1, P_2) = \mathrm{true}, I_c),$
$\quad \mathbf{intersect\_all}([I_a, I_c], I).$

SDF is satisfied iff we have: $\mathit{active}(P_1) = \mathrm{true} \wedge \mathit{close}(P_1, P_2) = \mathrm{true}$.

$\mathit{active}(P_1) = \mathrm{true}$ starts to hold while $\mathit{close}(P_1, P_2) = \mathrm{true}$ holds.
$\Rightarrow$ SDF gets satisfied.

Example: A Translatable SF

SF:

$\textbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\mathit{active}(P_1) = \text{true}), T),$
$\quad \textbf{holdsAt}(\mathit{close}(P_1, P_2) = \text{true}, T),$
$\quad \text{not } \textbf{happensAt}(\text{end}(\mathit{close}(P_1, P_2) = \text{true}), T).$

$\textbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\mathit{close}(P_1, P_2) = \text{true}), T),$
$\quad \textbf{holdsAt}(\mathit{active}(P_1) = \text{true}, T),$
$\quad \text{not } \textbf{happensAt}(\text{end}(\mathit{active}(P_1) = \text{true}), T).$

$\textbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\mathit{active}(P_1) = \text{true}), T),$
$\quad \textbf{happensAt}(\text{start}(\mathit{close}(P_1, P_2) = \text{true}), T).$

$\textbf{terminatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{end}(\mathit{active}(P_1) = \text{true}), T).$

$\textbf{terminatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{end}(\mathit{close}(P_1, P_2) = \text{true}), T).$

SDF:

$\textbf{holdsFor}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, I) \leftarrow$
$\quad \textbf{holdsFor}(\mathit{active}(P_1) = \text{true}, I_a),$
$\quad \textbf{holdsFor}(\mathit{close}(P_1, P_2) = \text{true}, I_c),$
$\quad \textbf{intersect\_all}([I_a, I_c], I).$

SDF is satisfied iff we have: $\mathit{active}(P_1) = \text{true} \wedge \mathit{close}(P_1, P_2) = \text{true}$.

$\mathit{active}(P_1) = \text{true}$ stops holding.
$\Rightarrow$ SDF gets violated.

Example: A Translatable SF

SF:

$\textbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\mathit{active}(P_1) = \text{true}), T),$
$\quad \textbf{holdsAt}(\mathit{close}(P_1, P_2) = \text{true}, T),$
$\quad \text{not } \textbf{happensAt}(\text{end}(\mathit{close}(P_1, P_2) = \text{true}), T).$

$\textbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\mathit{close}(P_1, P_2) = \text{true}), T),$
$\quad \textbf{holdsAt}(\mathit{active}(P_1) = \text{true}, T),$
$\quad \text{not } \textbf{happensAt}(\text{end}(\mathit{active}(P_1) = \text{true}), T).$

$\textbf{initiatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{start}(\mathit{active}(P_1) = \text{true}), T),$
$\quad \textbf{happensAt}(\text{start}(\mathit{close}(P_1, P_2) = \text{true}), T).$

$\textbf{terminatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{end}(\mathit{active}(P_1) = \text{true}), T).$

$\textbf{terminatedAt}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, T) \leftarrow$
$\quad \textbf{happensAt}(\text{end}(\mathit{close}(P_1, P_2) = \text{true}), T).$

SDF:

$\textbf{holdsFor}(\mathit{meeting}(P_1, P_2) = \mathit{interact}, I) \leftarrow$
$\quad \textbf{holdsFor}(\mathit{active}(P_1) = \text{true}, I_a),$
$\quad \textbf{holdsFor}(\mathit{close}(P_1, P_2) = \text{true}, I_c),$
$\quad \textbf{intersect\_all}([I_a, I_c], I).$

SDF is satisfied iff we have: $\mathit{active}(P_1) = \text{true} \wedge \mathit{close}(P_1, P_2) = \text{true}$.

SF includes 1 rule for each possible way of switching the truth value of: $\mathit{active}(P_1) = \text{true} \wedge \mathit{close}(P_1, P_2) = \text{true}$.
$\Rightarrow$ SF is inertial condition symmetric.

Translatable SFs

An SF is translatable to an SDF iff it is:

- inertial condition symmetric,
- guard condition symmetric and
- Boolean representation symmetric.

Theoretical Results

Translatable SFs

An SF is translatable to an SDF iff it is:

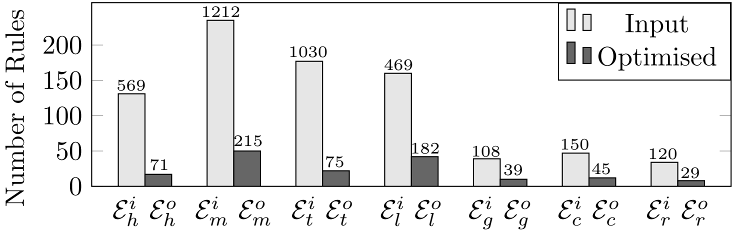
- inertial condition symmetric,
- guard condition symmetric and
- Boolean representation symmetric.

Compiler

We have devised and implemented an algorithm that:

- identifies the SFs that are translatable, and
- maps them into equivalent SDFs.

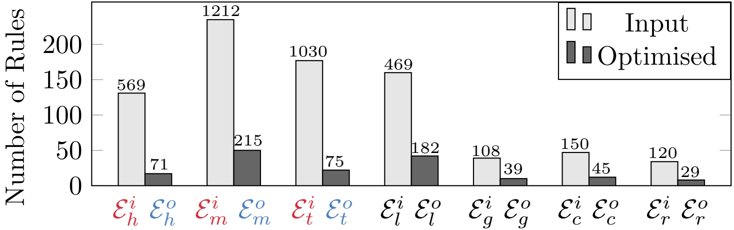
Experimental Evaluation

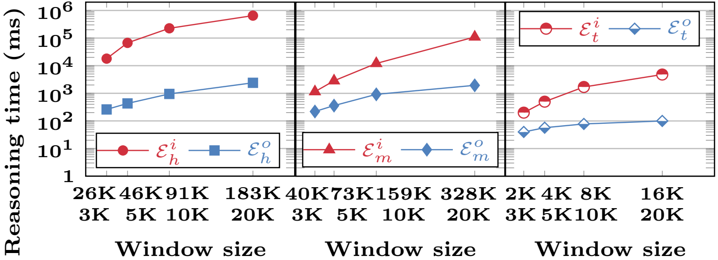


Event Calculus specifications for:

- human activity recognition.
- maritime situational awareness.
- city transport management.
- legal contract verification (Parvizimosaed et al. 2022).
- clinical guideline monitoring (Bragaglia et al. 2012).
- authorisation policy conflicts (Zahoor et al. 2022).
- redundant authorisation policies (Zahoor et al. 2023).

Experimental Evaluation





Code, Data & Temporal Specifications: https://github.com/aartikis/RTEC

Summary & Future Work

Summary:

- Formal characterisation of the class of SFs that are translatable into equivalent SDFs.

Summary & Future Work

Summary:

- Formal characterisation of the class of SFs that are translatable into equivalent SDFs.
- Compiler that identifies and re-writes them as SDFs.

Summary & Future Work

Summary:

- Formal characterisation of the class of SFs that are translatable into equivalent SDFs.
- Compiler that identifies and re-writes them as SDFs.
- Reproducible empirical evaluation on numerous real domain specifications.

Summary & Future Work

Summary:

- Formal characterisation of the class of SFs that are translatable into equivalent SDFs.
- Compiler that identifies and re-writes them as SDFs.
- Reproducible empirical evaluation on numerous real domain specifications.

Future Work:

- Compile RTEC specifications into automata, towards complex event forecasting[1].

[1] Alevizos et al., Complex event forecasting with prediction suffix tress. In VLDB Journal, 31(1), 157–180, 2022.