Scalable Relational Learning for Event Recognition

Doctoral Thesis Presentation

Nikos Katzouris

Department of Informatics & Telecommunications,
University of Athens

Institute of Informatics & Telecommunications,
NCSR “Demokritos”
Overview

Background
  Event Recognition & the Event Calculus
  Inductive Logic Programming

Incremental Learning Of Event Definitions
  Description of the Method
  Experimental Evaluation

Online Learning of Event Definitions
  Description of the Method
  Experimental Evaluation

Conclusions & Future Work
Background
Events

An event represents a time-evolving piece of information.

- Anything that happens/occurs.
  - E.g. a sensor signal, a transaction log etc.

- An event may be:
  - Instantaneous.
  - Durative.

- An event may be:
  - Simple event
  - Complex event.
Event Recognition

Simple Events

Recognition System

Complex Events

Complex Event Definitions

leaving object \((X, Y)\) at \(T\) if

- \(X\) is not inactive at \(T\) AND
- \(Y\) appears at \(T\) AND
- \(Y\) is inactive at \(T\) AND
- \(\text{coord}(X) = \text{coord}(Y)\) at \(T\)
Event Recognition

Input ▶

Simple Events

Recognition ▶

Event Recognition System

Complex Events

Output ▼

Complex Event Definitions

340 appear(id₀)
340 inactive(id₀)
340 coord(id₀)=(20.88, 11.90)
340 walking(id₂)
340 coord(id₂)=(25.88, 19.80)
340 active(id₁)
340 coord(id₁)=(20.88, 11.90)
340 walking(id₃)
340 coord(id₃)=(24.78, 18.77)
380 walking(id₃)
380 coord(id₃)=(27.88, 9.90)
380 walking(id₂)
380 coord(id₂)=(28.27, 9.66)
Event Recognition

Simple Events

- appear(id₀)
- inactive(id₀)
- coord(id₀) = (20.88, 11.90)
- walking(id₂)
- coord(id₂) = (25.88, 19.80)
- active(id₁)
- coord(id₁) = (20.88, 11.90)
- walking(id₃)
- coord(id₃) = (24.78, 18.77)
- walking(id₃)
- coord(id₃) = (27.88, 9.90)
- walking(id₂)
- coord(id₂) = (28.27, 9.66)

Complex Events

leaving_object(id₁, id₀)

leaving_object (X, Y) at T IFF
X is not inactive at T AND
Y appears at T AND
Y is inactive at T AND
 coord(X) equals coord(Y) at T

Complex Event Definitions
Event Recognition

Simple Events

340 appear(id₀)
340 inactive(id₀)
340 coord(id₀)=(20.88, 11.90)
340 walking(id₂)
340 coord(id₂)=(25.88, 19.80)
340 active(id₁)
340 coord(id₁)=(20.88, 11.90)
340 walking(id₃)
340 coord(id₃)=(24.78, 18.77)
380 walking(id₃)
380 coord(id₃)=(27.88, 9.90)
380 walking(id₂)
380 coord(id₂)=(28.27, 9.66)

Complex Events

340 leaving_object(id₁, id₀)
340 moving(id₂, id₃)
380 moving(id₂, id₃)

Complex Event Definitions

Event Recognition System

Input ➤

Recognition ➤

Output ➤
Logic-based Event Recognition

- Formal semantics.
- Relational structure.
- Connections to machine learning
  - Manual construction of event definitions is time consuming & error-prone
- Reasoning about time & change
  - Action formalisms.
### The Event Calculus

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>happensAt(E, T)</code></td>
<td>Event $E$ occurs at time $T$</td>
</tr>
<tr>
<td><code>initiatedAt(F, T)</code></td>
<td>At time $T$ a period of time for which fluent $F$ holds is initiated</td>
</tr>
<tr>
<td><code>terminatedAt(F, T)</code></td>
<td>At time $T$ a period of time for which fluent $F$ holds is terminated</td>
</tr>
<tr>
<td><code>holdsAt(F, T)</code></td>
<td>Fluent $F$ holds at time $T$</td>
</tr>
</tbody>
</table>

### Axioms

\[
\text{holdsAt}(F, T + 1) \leftarrow \text{initiatedAt}(F, T).
\]
\[
\text{holdsAt}(F, T + 1) \leftarrow \text{holdsAt}(F, T), \text{not terminatedAt}(F, T).
\]
Learning Complex Event Definitions

Simple Events → Event Recognition System → Complex Events

Learn this

Complex Event Definitions
Learning Complex Event Definitions

Input ▶

Simple Events

Recognition ▶

Event Recognition System

Complex Events

Output ■

Learn this

From These

Complex Event Definitions
Inductive Logic Programming

- **Input:**
  - Positive and negative examples.
  - Background knowledge.
  - Language bias.

- **Output:**
  - A logical theory that entails as many positive and as few negative examples as possible.
Inductive Logic Programming

Positive examples

Annotation:
holdsAt(meeting(id₁, id₂), 10)
holdsAt(meeting(id₂, id₁), 10)

Narrative:
happensAt(active(id₁), 9)
happensAt(active(id₂), 9)
coords(id₁, 86.5, 34.8, 9)
coords(id₂, 230.6, 176.4, 9)

Negative examples

Annotation:
not holdsAt(meeting(id₁, id₂), 20)
not holdsAt(meeting(id₂, id₁), 20)

Narrative:
happensAt(active(id₁), 19)
happensAt(active(id₂), 19)
coords(id₁, 140.5, 66.4, 19)
coords(id₂, 305.6, 190.5, 19)

Background Knowledge

Axioms of the Event Calculus

Language bias

head(initiatedAt(meeting(+person, +person), +time))

body(happensAt(active(+person), +time))
body(holdsAt(close(+person, +person, #distance), +time))
Learning a Clause: Structuring the Search Space

Top Clause $\top$:

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T).
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{inactive}(Y), T).
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T),
\]

\[
\text{happensAt}(\text{inactive}(Y), T),
\]

\[
\text{holdsAt}(\text{close}(X, Y, 25), T).
\]

\[
\ldots
\]

\[
\text{Rest of clauses generated by adding one condition from } \bot \text{ to the body of } \top.
\]

Bottom Clause $\bot$:

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T),
\]

\[
\text{happensAt}(\text{inactive}(Y), T),
\]

\[
\text{holdsAt}(\text{close}(X, Y, 25), T),
\]

\[
\text{holdsAt}(\text{close}(Y, X, 25), T),
\]

\[
\text{not happensAt}(\text{inactive}(X), T),
\]

\[
\text{not happensAt}(\text{active}(X), T),
\]

\[
\text{not happensAt}(\text{abrupt}(X), T),
\]

\[
\text{not happensAt}(\text{running}(X), T),
\]

\[
\text{happensAt}(\text{inactive}(Y), T),
\]

\[
\text{not happensAt}(\text{active}(Y), T),
\]

\[
\text{not happensAt}(\text{running}(Y), T),
\]

\[
\text{not happensAt}(\text{abrupt}(Y), T),
\]

\[
\text{holdsAt}(\text{orientation}(X, Y, 45), T).
\]
Learning a Clause: Structuring the Search Space

Top Clause \( \top \):
\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow
\]
\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T).
\]
\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{inactive}(Y), T).
\]
\[
\text{happensAt}(\text{active}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{holdsAt}(\text{close}(X, Y, 25), T).
\]

Bottom Clause \( \bot \):
\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{holdsAt}(\text{close}(X, Y, 25), T), \text{holdsAt}(\text{close}(Y, X, 25), T), \text{not happensAt}(\text{inactive}(X), T), \text{not happensAt}(\text{abrupt}(X), T), \text{not happensAt}(\text{running}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{not happensAt}(\text{active}(Y), T), \text{not happensAt}(\text{running}(Y), T), \text{not happensAt}(\text{abrupt}(Y), T), \text{holdsAt}(\text{orientation}(X, Y, 45), T).
\]

Rest of clauses generated by adding one condition from \( \bot \) to the body of \( \top \)

Ordered by \( \theta \)-subsumption \( \preceq \): If \( r_1 \preceq r_2 \) then \( \text{covers}(r_2, e) \Rightarrow \text{covers}(r_1, e) \) for all examples \( e \)
Covered examples

Learning a Clause: Search

Positives not covered

Top Clause $T$: \[ \text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \]

- \[ \text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{initiatedAt}(\text{meet}(X, Y), T). \]
- \[ \text{happensAt}(\text{active}(X), T). \]
- \[ \text{happensAt}(\text{inactive}(Y), T). \]

- \[ \text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{initiatedAt}(\text{meet}(X, Y), T). \]
- \[ \text{happensAt}(\text{active}(X), T). \]
- \[ \text{happensAt}(\text{inactive}(Y), T). \]

Covered examples

Bottom Clause $\bot$: \[ \text{happensAt}(\text{active}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{happensAt}(\text{close}(X, Y, 25), T). \]

- \[ \text{happensAt}(\text{active}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{happensAt}(\text{close}(X, Y, 25), T). \]
- \[ \text{not happensAt}(\text{inactive}(X), T), \text{not happensAt}(\text{abrupt}(X), T). \]
- \[ \text{not happensAt}(\text{running}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{not happensAt}(\text{active}(Y), T), \]
- \[ \text{not happensAt}(\text{running}(Y), T), \text{not happensAt}(\text{abrupt}(Y), T), \text{happensAt}(\text{orientation}(X, Y, 45), T). \]
Learning a Clause: Search

Generalization $\uparrow$

Top Clause $\top$

initiatedAt($meet(X, Y), T \leftarrow$

initiatedAt($meet(X, Y), T \leftarrow$

happensAt($active(X), T$),
happensAt($inactive(Y), T$).

Bottom Clause $\bot$

initiatedAt($meet(X, Y), T \leftarrow$

happensAt($active(X), T$),
happensAt($inactive(Y), T$),
holdsAt($close(X, Y, 25), T$).

initiatedAt($meet(X, Y), T \leftarrow$

happensAt($active(X), T$),
happensAt($inactive(Y), T$),
holdsAt($close(X, Y, 25), T$),
holdsAt($orientation(X, Y, 45), T$).

Learning a Clause: Search

**Negatives covered**

**Top Clause** \( T: \) initiatedAt(\( meet(X, Y), T \) \)

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T). \\
\text{happensAt}(\text{inactive}(Y), T).
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T). \\
\text{happensAt}(\text{inactive}(Y), T).
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T).
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T), \\
\text{happensAt}(\text{active}(Y), T).
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T), \\
\text{happensAt}(\text{active}(Y), T), \\
\text{holdsAt}(\text{close}(X, Y, 25), T).
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T), \\
\text{happensAt}(\text{active}(Y), T), \\
\text{holdsAt}(\text{close}(X, Y, 25), T).
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T), \\
\text{happensAt}(\text{active}(Y), T), \\
\text{not happensAt}(\text{close}(X, Y, 25), T).
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T), \\
\text{happensAt}(\text{active}(Y), T), \\
\text{not happensAt}(\text{close}(X, Y, 25), T), \\
\text{not happensAt}(\text{inactive}(X), T), \\
\text{not happensAt}(\text{active}(X), T), \\
\text{not happensAt}(\text{abrupt}(X), T), \\
\text{not happensAt}(\text{running}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T), \\
\text{not happensAt}(\text{active}(Y), T), \\
\text{not happensAt}(\text{running}(Y), T), \\
\text{not happensAt}(\text{abrupt}(Y), T), \\
\text{holdsAt}(\text{orientation}(X, Y, 45), T).
\]

**Bottom Clause** \( \bot: \)

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T), \\
\text{happensAt}(\text{active}(Y), T), \\
\text{holdsAt}(\text{close}(X, Y, 25), T).
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T), \\
\text{happensAt}(\text{active}(Y), T), \\
\text{not happensAt}(\text{close}(X, Y, 25), T).
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T), \\
\text{happensAt}(\text{active}(Y), T), \\
\text{not happensAt}(\text{close}(X, Y, 25), T), \\
\text{not happensAt}(\text{inactive}(X), T), \\
\text{not happensAt}(\text{active}(X), T), \\
\text{not happensAt}(\text{abrupt}(X), T), \\
\text{not happensAt}(\text{running}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T), \\
\text{not happensAt}(\text{active}(Y), T), \\
\text{not happensAt}(\text{running}(Y), T), \\
\text{not happensAt}(\text{abrupt}(Y), T), \\
\text{holdsAt}(\text{orientation}(X, Y, 45), T).
\]
Learning a Clause: Search

Specialization ↓

Top Clause T: \[\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow\]

\[\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{initiatedAt}(\text{meet}(X, Y), T)\]
\[\text{happensAt}(\text{active}(X), T)\]
\[\text{happensAt}(\text{inactive}(Y), T)\]

\[\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{initiatedAt}(\text{meet}(X, Y), T)\]
\[\text{happensAt}(\text{active}(X), T)\]
\[\text{happensAt}(\text{inactive}(Y), T)\]
\[\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{initiatedAt}(\text{meet}(X, Y), T)\]
\[\text{happensAt}(\text{active}(X), T)\]
\[\text{happensAt}(\text{inactive}(Y), T)\]

Bottom Clause ↓:

\[\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{initiatedAt}(\text{meet}(X, Y), T)\]
\[\text{happensAt}(\text{active}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{holdsAt}(\text{close}(X, Y, 25), T)\]
\[\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{initiatedAt}(\text{meet}(X, Y), T)\]
\[\text{happensAt}(\text{active}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{holdsAt}(\text{close}(X, Y, 25), T)\]
\[\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{initiatedAt}(\text{meet}(X, Y), T)\]
\[\text{happensAt}(\text{active}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{holdsAt}(\text{close}(X, Y, 25), T)\]
\[\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{initiatedAt}(\text{meet}(X, Y), T)\]
\[\text{happensAt}(\text{active}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{holdsAt}(\text{close}(X, Y, 25), T)\]
\[\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{initiatedAt}(\text{meet}(X, Y), T)\]
\[\text{happensAt}(\text{active}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{holdsAt}(\text{close}(X, Y, 25), T)\]

Covered examples:
Learning a Theory: Set-Cover Loop

\[ \mathcal{H} = \emptyset \ (\text{Begin}) \]

If there are positives not covered by \( \mathcal{H} \)?

- Return \( \mathcal{H} \)
- Select a positive example
  - Construct a Bottom Clause
  - Find a “good” clause \( r \):
    - SearchSpace: Clauses that \( \theta \)-subsume the bottom clause
    - \( \text{score}(r) \): A clause evaluation function
    - Return: \( r \in \text{SearchSpace} \) with the best score
- Covering step: Remove all positives covered by \( \mathcal{H} \)
  - \( \mathcal{H} = \mathcal{H} \cup r \)
  - Yes
  - Return \( \mathcal{H} \)

No
Learning Event Definitions with Classical ILP: Problems

\[
\text{holdsAt}(F, T + 1) \leftarrow \text{initiatedAt}(F, T).
\]

\[
\text{holdsAt}(F, T + 1) \leftarrow \text{holdsAt}(F, T) \leftarrow \text{not terminatedAt}(F, T).
\]

- Because of Negation as Failure:
  - Set-cover doesn’t work.
  - Horn logic subsumption-based heuristics don’t work.
  - A **sound** theory cannot be constructed by learning clauses in isolation.
Learning Event Definitions with Classical ILP: Problems

\[
\text{holdsAt}(F, T + 1) \leftarrow \\
\quad \text{initiatedAt}(F, T).
\]

\[
\text{holdsAt}(F, T + 1) \leftarrow \\
\quad \text{holdsAt}(F, T) \leftarrow, \\
\quad \text{not} \ \text{terminatedAt}(F, T).
\]

- **Because of Negation as Failure:**
  - Set-cover doesn’t work.
  - Horn logic subsumption-based heuristics don’t work.
  - A **sound** theory cannot be constructed by learning clauses in isolation.

- **Non-Observational Predicate Learning (non-OPL):**
  - Annotation predicates differ from target predicates.
Abductive-Inductive Logic Programming

Abductive Logic Programming (ALP)
- Reasoning for explaining observations.
  - A complex event holds at some time...
  - Therefore it must have been initiated previously.
  - **Means to solve non-OPL.**
- ALP has a non-monotonic semantics.
  - **Overcome problems related to Negation as Failure.**
ILP (Overview)

Classical ILP systems
- Designed for Horn logic.
- OPL-only.

Non-monotonic ILP systems
- ALP + ILP.
- Designed for full-clausal logic.
- Support non-OPL.
- Capable of Theory Revision.
- Learn whole theories.

More advanced ILP systems
- Designed for Horn logic.
- Support non-OPL.

Classical Theory Revision systems
- Designed for Horn logic.
- OPL-only.
The XHAIL System

Examples:

\(\text{happensAt}(\text{walking}(id_1), 9)\)
\(\text{coords}(id_1, 201, 454, 9)\)
\(\text{holdsAt}(\text{moving}(id_1, id_2), 10)\)
...
\(\text{happensAt}(\text{running}(id_5), 20)\)
\(\text{direction}(id_3, 270, 20)\)
\(\text{not holdsAt}(\text{moving}(id_3, id_5), 21)\)
...

Abduction:

\(\text{initiatedAt}(\text{moving}(id_1, id_2), 9)\)
\(\text{terminatedAt}(\text{moving}(id_3, id_5), 20)\)
The XHAIL System

Examples:

happensAt(walking(id₁), 9)
coords(id₁, 201, 454, 9)
holdsAt(moving(id₁, id₂), 10)
...
happensAt(running(id₅), 20)
direction(id₃, 270, 20)
not holdsAt(moving(id₃, id₅), 21)
...

Abduction:

initiatedAt(moving(id₁, id₂), 9)
terminatedAt(moving(id₃, id₅), 20)

Kernel Set:

initiatedAt(moving(id₁, id₂), 9) ←
happensAt(walking(id₁), 9),
happensAt(walking(id₂), 9),
not happensAt(running(id₁), 9),
not happensAt(active(id₂), 9),
holdsAt(close(id₁, id₂, 35), 9),
holdsAt(orientation(id₁, id₂, 45), 9).

terminatedAt(moving(id₃, id₅), 20) ←
happensAt(walking(id₃), 20),
happensAt(running(id₅), 20),
not happensAt(abrupt(id₁), 9),
not happensAt(active(id₂), 9),
not holdsAt(close(id₁, id₂, 35), 20),
not holdsAt(orientation(id₁, id₂, 45), 20).
The XHAIL System

**Examples:**
- `happensAt(walking(id₁), 9)`
- `coords(id₁, 201, 454, 9)`
- `holdsAt(moving(id₁, id₂), 10)`
- `happensAt(running(id₅), 20)`
- `direction(id₃, 270, 20)`
- `not holdsAt(moving(id₃, id₅), 21)`

**Abduction:**
- `initiatedAt(moving(id₁, id₂), 9)`
- `terminatedAt(moving(id₃, id₅), 20)`

**Kernel Set:**
- `initiatedAt(moving(X, Y), T) ← happensAt(walking(X), T), happensAt(walking(Y), T), holdsAt(close(X, Y, 35), T), not happensAt(running(X), T), not happensAt(active(Y), T), holdsAt(orientation(X, Y, 45), T).`
- `terminatedAt(moving(X, Y), T) ← happensAt(running(Y), T), not happensAt(active(Y), T), not happensAt(abrupt(X), T), not happensAt(active(Y), T), not holdsAt(close(X, Y, 35), T), not holdsAt(orientation(X, Y, 45), T).`
The XHAIL System

**Examples:**
- `happensAt(walking(id_1), 9)`
- `coords(id_1, 201, 454, 9)`
- `holdsAt(moving(id_1, id_2), 10)`

...`happensAt(running(id_5), 20)`
- `direction(id_3, 270, 20)`
- `not holdsAt(moving(id_3, id_5), 21)`

**Abduction:**
- `initiatedAt(moving(id_1, id_2), 9)`
- `terminatedAt(moving(id_3, id_5), 20)`

**Kernel Set:**
- `initiatedAt(moving(X, Y), T) ← happensAt(walking(X), T), happensAt(walking(Y), T), holdsAt(close(X, Y, 35), T), not happensAt(running(X), T), not happensAt(active(Y), T), holdsAt(orientation(X, Y, 45), T).`

- `terminatedAt(moving(X, Y), T) ← happensAt(walking(X), T), happensAt(running(Y), T), not happensAt(abrupt(X), T), not happensAt(active(Y), T), not holdsAt(close(X, Y, 35), T), not holdsAt(orientation(X, Y, 45), T).`

**Hypothesis:**
- `initiatedAt(moving(X, Y), T) ← happensAt(walking(X), T), happensAt(walking(X), T), holdsAt(close(X, Y, 35), T), holdsAt(orientation(X, Y, 45), T), terminatedAt(moving(X, Y), T) ← happensAt(running(Y), T), not holdsAt(close(X, Y, 35), T).`
Incremental Learning Of Event Definitions
Building on XHAIL

The XHAIL system:

- Is able to learn sound theories in the Event Calculus.
- But does not scale.
  - Learns whole theories from the entirety of training examples.

Goal:

- Learn incrementally (examples arrive over time).
- Revise past theories to fit new examples.
- Full memory.
- Preserve soundness.
**Incremental Learning**

<table>
<thead>
<tr>
<th></th>
<th>ok!</th>
<th>ok!</th>
<th>ok!</th>
<th>ok!</th>
<th>not ok!</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image-url" alt="Green Circle" /></td>
<td><img src="image-url" alt="Green Circle" /></td>
<td><img src="image-url" alt="Green Circle" /></td>
<td><img src="image-url" alt="Green Circle" /></td>
<td><img src="image-url" alt="Red Circle" /></td>
</tr>
</tbody>
</table>

We must start all over again.
Incremental Learning

1. ok! ok! ok! ok! not ok!


3. ? ? ok! ok!

4. ? ok! ok! ok!

5. ? not ok! ok! ok!

6. ? ok! ok! ?

We must start all over again.
Incremental Learning

1. ok! ok! ok! ok! not ok!


3. ? ? .... ? ok! ok!
Incremental Learning

1. **ok!** | **ok!** | ...... | **ok!** | **ok!** | **not ok!**
2. **?** | **?** | ...... | **?** | **?** | **ok!**
3. **?** | **?** | ...... | **?** | **ok!** | **ok!**
4. **?** | **?** | ...... | **ok!** | **ok!** | **ok!**
5. **?** | **?** | ...... | **ok!** | **ok!** | **ok!**
Incremental Learning

We must start all over again.
We must start all over again
Support Set

\[
\begin{align*}
\text{initiatedAt(meeting}(X, Y), T) & \leftarrow \text{happensAt(active}(X), T), \\
& \quad \text{happensAt(active}(Y), T), \\
& \quad \text{holdsAt(close}(X, Y, 25), T).
\end{align*}
\]

\[
\begin{align*}
\text{terminatedAt(meeting}(X, Y), T) & \leftarrow \text{happensAt(walking}(X), T), \\
& \quad \text{happensAt(running}(Y), T), \\
& \quad \text{not happensAt(close}(X, Y, 25), T).
\end{align*}
\]

\[
\begin{align*}
\text{initiatedAt(meeting}(X, Y), T) & \leftarrow \text{happensAt(active}(X), T), \\
& \quad \text{happensAt(inactive}(Y), T), \\
& \quad \text{holdsAt(close}(X, Y, 25), T), \\
& \quad \text{not happensAt(running}(X), T), \\
& \quad \text{not holdsAt(orientation}(X, Y, 45), T), \\
& \quad \text{holdsAt(close}(Y, X, 25), T).
\end{align*}
\]

\[
\begin{align*}
\text{initiatedAt(meeting}(X, Y), T) & \leftarrow \text{happensAt(active}(X), T), \\
& \quad \text{happensAt(inactive}(Y), T), \\
& \quad \text{holdsAt(close}(X, Y, 25), T), \\
& \quad \text{not happensAt(running}(X), T), \\
& \quad \text{not holdsAt(orientation}(X, Y, 45), T), \\
& \quad \text{not happensAt(running}(X), T).
\end{align*}
\]

\[
\begin{align*}
\text{terminatedAt(meeting}(X, Y), T) & \leftarrow \text{happensAt(walking}(X), T), \\
& \quad \text{happensAt(running}(Y), T), \\
& \quad \text{not happensAt(abrupt}(X), T), \\
& \quad \text{not happensAt(active}(Y), T), \\
& \quad \text{not holdsAt(close}(X, Y, 25), T), \\
& \quad \text{not holdsAt(close}(Y, X, 25), T), \\
& \quad \text{not happensAt(inactive}(Y), T).
\end{align*}
\]
Support Set

\[ \text{initiatedAt}(\text{meeting}(X, Y), T) \leftarrow \]
\[ \text{happensAt}(\text{active}(X), T), \]
\[ \text{happensAt}(\text{active}(Y), T), \]
\[ \text{holdsAt}(\text{close}(X, Y, 25), T). \]

\[ \theta\text{-subsumes} \]

\[ \text{initiatedAt}(\text{meeting}(X, Y), T) \leftarrow \]
\[ \text{happensAt}(\text{active}(X), T), \]
\[ \text{happensAt}(\text{active}(Y), T), \]
\[ \text{holdsAt}(\text{close}(X, Y, 25), T), \]
\[ \text{not happensAt}(\text{running}(X), T), \]
\[ \text{not happensAt}(\text{inactive}(Y), T), \]
\[ \text{holdsAt}(\text{orientation}(X, Y, 45), T), \]
\[ \text{holdsAt}(\text{close}(Y, X, 25), T). \]

t_{n-1}
Support Set

\[\text{initiatedAt}(\text{meeting}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \]
\[\text{holdsAt}(\text{close}(X, Y, 25), T).\]

\[\theta\text{-subsumes}\]

\[\text{initiatedAt}(\text{meeting}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \]
\[\text{happensAt}(\text{active}(Y), T), \]
\[\text{holdsAt}(\text{close}(X, Y, 25), T), \]
\[\text{not happensAt}(\text{running}(X), T), \]
\[\text{not happensAt}(\text{inactive}(Y), T), \]
\[\text{holdsAt}(\text{orientation}(X, Y, 45), T), \]
\[\text{holdsAt}(\text{close}(Y, X, 25), T).\]

\[\theta\text{-subsumes}\]

\[\text{initiatedAt}(\text{meeting}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \]
\[\text{holdsAt}(\text{close}(X, Y, 25), T).\]

\[\text{covers}\]

\[t_{n-1}\]

\[t_n\]
Support Set

\[ \text{initiatedAt}(\text{meeting}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \text{happensAt}(\text{active}(Y), T), \text{holdsAt}(\text{close}(X, Y, 25), T). \]

\[ \theta \text{-subsumes} \]

\[ \text{initiatedAt}(\text{meeting}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \text{happensAt}(\text{active}(Y), T), \text{holdsAt}(\text{close}(X, Y, 25), T), \text{not happensAt}(\text{running}(X), T), \text{not happensAt}(\text{inactive}(Y), T), \text{holdsAt}(\text{orientation}(X, Y, 45), T), \text{holdsAt}(\text{close}(Y, X, 25), T). \]

\[ \text{covers} \]

\[ t_{n-1} \quad t_n \quad t_{n+1} \]
\[ \text{initiatedAt}(\text{meeting}(X, Y), T) \leftarrow \]
\[ \text{happensAt}(\text{active}(X), T), \]
\[ \text{happensAt}(\text{active}(Y), T), \]
\[ \text{happensAt}(\text{close}(X, Y, 25), T). \]

\[ \theta\text{-subsumes} \]

\[ \text{initiatedAt}(\text{meeting}(X, Y), T) \leftarrow \]
\[ \text{happensAt}(\text{active}(X), T), \]
\[ \text{happensAt}(\text{active}(Y), T), \]
\[ \text{happensAt}(\text{close}(X, Y, 25), T), \]
\[ \text{not happensAt}(\text{running}(X), T), \]
\[ \text{not happensAt}(\text{inactive}(Y), T), \]
\[ \text{holdsAt}(\text{orientation}(X, Y, 45), T), \]
\[ \text{holdsAt}(\text{close}(Y, X, 25), T). \]

\[ \theta\text{-subsumes} \]

\[ \text{initiatedAt}(\text{meeting}(X, Y), T) \leftarrow \]
\[ \text{happensAt}(\text{active}(X), T), \]
\[ \text{happensAt}(\text{active}(Y), T), \]
\[ \text{happensAt}(\text{close}(X, Y, 25), T), \]
\[ \text{not happensAt}(\text{active}(Y), T), \]
\[ \text{holdsAt}(\text{close}(X, Y, 25), T), \]
\[ \text{holdsAt}(\text{orientation}(X, Y, 45), T), \]
\[ \text{not happensAt}(\text{running}(X), T). \]
initiatedAt(meeting(X, Y), T) ← happensAt(active(X), T),
holdsAt(close(X, Y, 25), T).

θ-subsumes

initiatedAt(meeting(X, Y), T) ← happensAt(active(X), T),
happensAt(active(Y), T),
holdsAt(close(X, Y, 25), T),
not happensAt(running(X), T),
not happensAt(inactive(Y), T),
holdsAt(orientation(X, Y, 45), T),
holdsAt(close(Y, X, 25), T).

θ-subsumes

initiatedAt(meeting(X, Y), T) ← happensAt(active(X), T),
happensAt(inactive(Y), T),
holdsAt(close(X, Y, 25), T),
holdsAt(close(Y, X, 25), T),
not happensAt(active(Y), T),
holdsAt(orientation(X, Y, 45), T),
not happensAt(running(X), T).

covers

t_{n−1}

t_{n}

t_{n+1}

covers

t_{n+2}

t_{n+3}
initiatedAt(meeting(X, Y), T) ←
happensAt(active(X), T),
happensAt(active(Y), T),
happensAt(close(X, Y, 25), T),
not happensAt(running(X), T),
not happensAt(inactive(Y), T),
happensAt(orientation(X, Y, 45), T),
happensAt(close(Y, X, 25), T).

θ-subsumes

θ-subsumes

initiatedAt(meeting(X, Y), T) ←
happensAt(active(X), T),
happensAt(inactive(Y), T),
happensAt(close(X, Y, 25), T),
not happensAt(running(X), T),
not happensAt(inactive(Y), T),
happensAt(orientation(X, Y, 45), T),
happensAt(close(Y, X, 25), T),
not happensAt(running(X), T).

covers!

covers!
If a specialization $\theta$-subsumes the support set...

$\theta$-subsumes

$\theta$-subsumes

...it preserves previously-covered positives.
Scalable Theory Revision

- Revision operators:
  - Clause addition.
  - Clause specialization.

- When example windows arrive over time:
  - At most one pass over the history memory for each new window.
  - If the revision involves specializations only, no look-back.
  - If the revision involves clause addition, single pass.

- When example windows are in place from the start:
  - Two passes over the data suffice:
    - Cover all positives in the first pass.
    - Reject any negatives in the second pass.

- Total cost depends on mean unit cost for one window.
  - May be kept low for low window sizes.
Scalable Theory Revision

- Revision operators:
  - Clause addition.
  - Clause specialization.

- When example windows arrive over time:
  - At most one pass over the history memory for each new window.
    - If the revision involves specializations only, no look-back.
    - If the revision involves clause addition, single pass.

- When example windows are in place from the start:
  - Two passes over the data suffice:
    - Cover all positives in the first pass.
    - Reject any negatives in the second pass.

- Total cost depends on mean unit cost for one window.
  - May be kept low for low window sizes.
Scalable Theory Revision

- Revision operators:
  - Clause addition.
  - Clause specialization.

- When example windows arrive over time:
  - At most one pass over the history memory for each new window.
    - If the revision involves specializations only, no look-back.
    - If the revision involves clause addition, single pass.

- When example windows are in place from the start:
  - Two passes over the data suffice:
    - Cover all positives in the first pass.
    - Reject any negatives in the second pass.

Total cost depends on mean unit cost for one window. May be kept low for low window sizes.
Scalable Theory Revision

▸ Revision operators:
  ▸ Clause addition.
  ▸ Clause specialization.

▸ When example windows arrive over time:
  ▸ At most one pass over the hist. memory for each new window.
    ▸ If the revision involves specializations only, no look-back.
    ▸ If the revision involves clause addition, single pass.

▸ When example windows are in place from the start:
  ▸ Two passes over the data suffice:
    ▸ Cover all positives in the first pass.
    ▸ Reject any negatives in the second pass.

▸ Total cost depends on mean unit cost for one window.
  ▸ May be kept low for low window sizes.
Experimental Evaluation: Activity Recognition

- Experiments with the CAVIAR dataset for activity recognition.
  - Small fragment of real data (hand-picked).
  - Large synthetic dataset.
    - Noise-free, but more complex than the real dataset.
## Experimental Evaluation: Activity Recognition

<table>
<thead>
<tr>
<th>Real CAVIAR data</th>
<th>ILED</th>
<th>XHAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G = 10 )</td>
<td>34.15</td>
<td>1560.88</td>
</tr>
<tr>
<td>( G = 50 )</td>
<td>23.04</td>
<td>–</td>
</tr>
<tr>
<td>( G = 100 )</td>
<td>286.74</td>
<td>15</td>
</tr>
<tr>
<td>( G = 900 )</td>
<td>–</td>
<td>17.5</td>
</tr>
<tr>
<td>Training Time (sec)</td>
<td>34.15</td>
<td>1560.88</td>
</tr>
<tr>
<td>Revisions</td>
<td>11.2</td>
<td>–</td>
</tr>
<tr>
<td>Hypothesis size</td>
<td>17.82</td>
<td>15</td>
</tr>
<tr>
<td>Precision</td>
<td>98.713</td>
<td>99.973</td>
</tr>
<tr>
<td>Recall</td>
<td>99.789</td>
<td>99.992</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Synthetic CAVIAR data</th>
<th>ILED</th>
<th>XHAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G = 10 )</td>
<td>38.92</td>
<td>21429</td>
</tr>
<tr>
<td>( G = 50 )</td>
<td>33.87</td>
<td>–</td>
</tr>
<tr>
<td>( G = 100 )</td>
<td>468</td>
<td>118.18</td>
</tr>
<tr>
<td>( G = 1000 )</td>
<td>–</td>
<td>63.822</td>
</tr>
<tr>
<td>Training Time (sec)</td>
<td>38.92</td>
<td>21429</td>
</tr>
<tr>
<td>Revisions</td>
<td>28.7</td>
<td>–</td>
</tr>
<tr>
<td>Hypothesis size</td>
<td>143.52</td>
<td>118.18</td>
</tr>
<tr>
<td>Precision</td>
<td>55.713</td>
<td>63.822</td>
</tr>
<tr>
<td>Recall</td>
<td>68.213</td>
<td>71.918</td>
</tr>
</tbody>
</table>
Experimental Evaluation: Transport Management

- Data from sensors installed on trams & buses.
  - Changes in position.
  - Acceleration/deceleration.
  - In-vehicle temperature.
  - Noise level and passenger density.

- Complex events related to:
  - Punctuality of a public transport vehicle.
  - Passenger/driver comfort and safety.
Complex events form a hierarchy:
Attempt 1: Learn definitions for all concepts simultaneously:

<table>
<thead>
<tr>
<th></th>
<th>ILED</th>
<th>XHAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$G = 5$</td>
<td>$G = 10$</td>
</tr>
<tr>
<td>Training Time (hours)</td>
<td>1.35</td>
<td>1.88</td>
</tr>
<tr>
<td>Hypothesis size</td>
<td>28.32</td>
<td>24.13</td>
</tr>
<tr>
<td>Revisions</td>
<td>14.78</td>
<td>13.42</td>
</tr>
<tr>
<td>Precision</td>
<td>63.344</td>
<td>64.644</td>
</tr>
<tr>
<td>Recall</td>
<td>59.832</td>
<td>61.423</td>
</tr>
</tbody>
</table>
### Attempt 2: Learning with hierarchical bias:

<table>
<thead>
<tr>
<th>Level</th>
<th>G = 10 (min)</th>
<th>G = 50 (min)</th>
<th>G = 100 (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Level-1</strong></td>
<td>4.46 – 4.88</td>
<td>5.78 – 6.44</td>
<td>6.24 – 6.88</td>
</tr>
<tr>
<td>Training Time</td>
<td>4.46 – 4.88</td>
<td>5.78 – 6.44</td>
<td>6.24 – 6.88</td>
</tr>
<tr>
<td>Revisions</td>
<td>2 – 11</td>
<td>2 – 9</td>
<td>2 – 9</td>
</tr>
<tr>
<td>Hypothesis size</td>
<td>4 – 18</td>
<td>4 – 16</td>
<td>4 – 16</td>
</tr>
<tr>
<td>Precision</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Recall</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Level-2</strong></td>
<td>8.76</td>
<td>9.14</td>
<td>9.86</td>
</tr>
<tr>
<td>Training Time</td>
<td>8.76</td>
<td>9.14</td>
<td>9.86</td>
</tr>
<tr>
<td>Revisions</td>
<td>24</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Hypothesis size</td>
<td>31</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>Precision</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Recall</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Level-3</strong></td>
<td>5.78</td>
<td>6.14</td>
<td>6.78</td>
</tr>
<tr>
<td>Training Time</td>
<td>5.78</td>
<td>6.14</td>
<td>6.78</td>
</tr>
<tr>
<td>Revisions</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Hypothesis size</td>
<td>13</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Precision</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Recall</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>
Online Learning of Event Definitions
Challenges

- Soundness is not always useful.
- Maintaining a full memory is not always possible.
  - Streaming data cannot be stored.
Challenges

▶ Soundness is not always useful.
▶ Maintaining a full memory is not always possible.
  ▶ Streaming data cannot be stored.

Goal:
▶ Noise-tolerant learning in an online fashion.
  ▶ Examples arrive in a stream.
  ▶ Single-pass learners: Each example is “seen” once.
Challenges

- Soundness is not always useful.
- Maintaining a full memory is not always possible.
  - Streaming data cannot be stored.

Goal:
- Noise-tolerant learning in an online fashion.
  - Examples arrive in a stream.
  - Single-pass learners: Each example is “seen” once.

Approach:
- Make decisions from subsets of the stream:
  - Decisions are optimal “locally”.
  - Decisions are optimal “globally”...
    - within an error margin $\epsilon$.
    - with probability $1 - \delta$. 
The Hoeffding Bound

- $X$ is a random variable
- $X_1, \ldots, X_N$ are $N$ independent observations of $X$’s values
- Let $\bar{X}$ be the known, observed mean of $X$.
- Let $\hat{X}$ be the unknown, true mean of $X$.
- It holds that:

$$\bar{X} - \epsilon \leq \hat{X} \leq \bar{X} + \epsilon,$$
with probability $1 - \delta$, where

$$\epsilon = \sqrt{\frac{\ln(1/\delta)}{2N}}$$
Learning a Clause in an Online Fashion

Candidate clauses

\[ R_1: 0.345 \]

\[ R_2: 0.232 \]

\[ R_3: 0.145 \]

\[ R_4: 0.612 \]

\[ R_5: 0.325 \]

Training stream

Find the best candidate across the stream

\[ \bar{X} - \epsilon \leq \hat{X} \leq \bar{X} + \epsilon, \] where \[ \epsilon = \sqrt{\ln \left( \frac{1}{\delta} \right) / 2N} \]

Find the best candidate across the stream

Training stream

\[ \bar{X} - \epsilon \leq \hat{X} \leq \bar{X} + \epsilon, \] where \[ \epsilon = \sqrt{\ln \left( \frac{1}{\delta} \right) / 2N} \]

\[ \hat{X} \geq \bar{X} - \epsilon > 0 \]

\[ \Rightarrow \text{score BestClause} - \text{score SecondBestClause} > 0 \]

\[ \Rightarrow \text{BestClause} \text{ is indeed the best clause, with probability } 1 - \delta. \]
Learning a Clause in an Online Fashion

Candidate clauses

- $R_1: 0.345$
- $R_2: 0.232$
- $R_3: 0.145$
- $R_4: 0.612$
- $R_5: 0.325$

Training stream

Find the best candidate across the stream

As examples stream in...

Monitor $\bar{X} = score_{BestClause} - score_{SecondBestClause}$

Then $\hat{X} \geq \bar{X} - \epsilon > 0 \Rightarrow score_{BestClause} - score_{SecondBestClause} > 0 \Rightarrow BestClause$ is indeed the best clause, with probability $1 - \delta$. 
Learning a Clause in an Online Fashion

Candidate clauses

\[ R_1: 0.345 \]
\[ R_2: 0.232 \]
\[ R_3: 0.145 \]
\[ R_4: 0.612 \]
\[ R_5: 0.325 \]

Find the best candidate across the stream

Training stream

\[ \bar{X} - \epsilon \leq \hat{X} \leq \bar{X} + \epsilon, \] where \[ \epsilon = \sqrt{\ln(1/\delta) / 2N} \]

Candidate clauses

As examples stream in...

\[ \text{Monitor } \bar{X} = \text{score}_{\text{BestClause}} - \text{score}_{\text{SecondBestClause}} \]

Continue until the number \( N \) of examples

\[ \text{Makes } \bar{X} > \epsilon = \sqrt{\ln(1/\delta) / 2N} \]
Learning a Clause in an Online Fashion

Find the best candidate across the stream

Candidate clauses

- $R_1: 0.345$
- $R_2: 0.232$
- $R_3: 0.145$
- $R_4: 0.612$
- $R_5: 0.325$

As examples stream in...

Monitor $\hat{X} = \text{score}_{\text{BestClause}} - \text{score}_{\text{SecondBestClause}}$

Continue until the number $N$ of examples

Makes $\hat{X} > \epsilon = \sqrt{\frac{\ln(1/\delta)}{2N}}$

Then

$\hat{X} \geq \hat{X} - \epsilon > 0 \Rightarrow$

$\text{score}_{\text{BestClause}} - \text{score}_{\text{SecondBestClause}} > 0 \Rightarrow$

BestClause is indeed the best clause, with probability $1 - \delta$. 

$\bar{X} - \epsilon \leq \hat{X} \leq \bar{X} + \epsilon$, where $\epsilon = \sqrt{\frac{\ln(1/\delta)}{2N}}$
Online Hill-Climbing

Top Clause $\top$:

\[ \text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \]

\[ \text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T). \]

\[ \text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{inactive}(Y), T). \]

\[ \cdots \]

\[ \text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{orientation}(X, Y, 45), T). \]

Bottom Clause $\bot$:

\[ \cdots \]

\[ \cdots \]

\[ \cdots \]

\[ \text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{holdsAt}(\text{close}(X, Y, 25), T), \]

\[ \text{holdsAt}(\text{close}(Y, X, 25), T), \text{not happensAt}(\text{inactive}(X), T), \text{not happensAt}(\text{abrupt}(X), T), \]

\[ \text{not happensAt}(\text{running}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{not happensAt}(\text{active}(Y), T), \]

\[ \text{not happensAt}(\text{running}(Y), T), \text{not happensAt}(\text{abrupt}(Y), T), \text{holdsAt}(\text{orientation}(X, Y, 45), T). \]
Online Hill-Climbing

Bottom Clause $\bot$:

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \\
\text{happensAt}(\text{active}(X), T) \\
\text{happensAt}(\text{inactive}(Y), T). \\
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \\
\text{happensAt}(\text{active}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T), \\
\text{happensAt}(\text{close}(X, Y, 25), T), \\
\text{happensAt}(\text{close}(Y, X, 25), T), \\
\text{not happensAt}(\text{inactive}(X), T), \\
\text{not happensAt}(\text{active}(X), T), \\
\text{not happensAt}(\text{running}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T), \\
\text{not happensAt}(\text{active}(Y), T), \\
\text{not happensAt}(\text{running}(Y), T), \\
\text{not happensAt}(\text{abrupt}(X), T), \\
\text{not happensAt}(\text{abrupt}(Y), T), \\
\text{holdsAt}(\text{orientation}(X, Y, 45), T). \\
\]

Training stream

Best clause so far

Used $\mathcal{O}\left(\frac{1}{\epsilon^2 \ln \frac{1}{\delta}}\right)$ examples
Online Hill-Climbing

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T).
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{inactive}(Y), T).
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{inactive}(X), T), \text{holdsAt}(\text{close}(X, Y, 25), T).
\]

\[
\text{Bottom Clause } \perp:
\]

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \leftarrow \text{happensAt}(\text{active}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{holdsAt}(\text{close}(X, Y, 25), T), \text{holdsAt}(\text{close}(Y, X, 25), T), \text{not happensAt}(\text{inactive}(X), T), \text{not happensAt}(\text{inactive}(Y), T), \text{not happensAt}(\text{abrupt}(X), T), \text{not happensAt}(\text{running}(X), T), \text{happensAt}(\text{inactive}(Y), T), \text{not happensAt}(\text{active}(Y), T), \text{not happensAt}(\text{running}(Y), T), \text{not happensAt}(\text{abrupt}(Y), T), \text{holdsAt}(\text{orientation}(X, Y, 45), T).
\]

\[
\text{Training stream}
\]

\[
\text{Used } \mathcal{O}(\frac{1}{\epsilon^2} \ln \frac{1}{\delta}) \text{ examples}
\]
Online Hill-Climbing

initiatedAt(meet(X, Y), T) ←

initiatedAt(meet(X, Y), T) ←
  initiatedAt(meet(X, Y), T).

initiatedAt(meet(X, Y), T) ←
  initiatedAt(meet(X, Y), T) ←
    initiatedAt(meet(X, Y), T) ←
      initiatedAt(meet(X, Y), T) ←
        initiatedAt(meet(X, Y), T) ←
          initiatedAt(meet(X, Y), T) ←
            initiatedAt(meet(X, Y), T).

happensAt(active(X), T).

happensAt(inactive(Y), T).

holdsAt(close(X, Y, 25), T).

holdsAt(orientation(X, Y, 45), T).

initiatedAt(meet(X, Y), T) ←
  initiatedAt(meet(X, Y), T) ←
    initiatedAt(meet(X, Y), T) ←
      initiatedAt(meet(X, Y), T) ←
        initiatedAt(meet(X, Y), T) ←
          initiatedAt(meet(X, Y), T) ←
            initiatedAt(meet(X, Y), T).

happensAt(active(X), T).

happensAt(inactive(Y), T).

holdsAt(active(X), T).

holdsAt(inactive(Y), T).

holdsAt(close(X, Y, 25), T).

holdsAt(orientation(X, Y, 45), T).

Bottom Clause ⊥:

initiatedAt(meet(X, Y), T) ←
  initiatedAt(meet(X, Y), T) ←
    initiatedAt(meet(X, Y), T) ←
      initiatedAt(meet(X, Y), T) ←
        initiatedAt(meet(X, Y), T) ←
          initiatedAt(meet(X, Y), T) ←
            initiatedAt(meet(X, Y), T).

happensAt(active(X), T).

happensAt(inactive(Y), T), holdsAt(close(X, Y, 25), T), holdsAt(close(Y, X, 25), T), not happensAt(inactive(Y), T), not happensAt(abrupt(X), T), not happensAt(abrupt(Y), T), not happensAt(abrupt(Y), T), not happensAt(abrupt(Y), T), holdsAt(orientation(X, Y, 45), T).

Training stream

Used $O\left(\frac{1}{\epsilon^2 \ln \frac{1}{\delta}}\right)$ examples

Used $O\left(\frac{1}{\epsilon^2 \ln \frac{1}{\delta}}\right)$ examples

Best clause so far

Bottom Clause ⊥:
Online Hill-Climbing

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \gets \\
\text{happensAt}(\text{active}(X), T). \\
\text{initiatedAt}(\text{meet}(X, Y), T) \gets \\
\text{happensAt}(\text{inactive}(Y), T). \\
\text{initiatedAt}(\text{meet}(X, Y), T) \gets \\
\text{holdsAt}(\text{close}(X, Y, 25), T).
\]

Bottom Clause \( \bot \):

\[
\text{initiatedAt}(\text{meet}(X, Y), T) \gets \\
\text{happensAt}(\text{active}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T), \\
\text{holdsAt}(\text{close}(X, Y, 25), T), \\
\text{not happensAt}(\text{inactive}(X), T), \\
\text{not happensAt}(\text{abrupt}(X), T), \\
\text{not happensAt}(\text{running}(X), T), \\
\text{happensAt}(\text{inactive}(Y), T), \\
\text{not happensAt}(\text{active}(Y), T), \\
\text{not happensAt}(\text{running}(Y), T), \\
\text{not happensAt}(\text{abrupt}(Y), T), \\
\text{holdsAt}(\text{orientation}(X, Y, 45), T).
\]

Training stream

\[\text{Used } \mathcal{O}(\frac{1}{\epsilon^2} \ln \frac{1}{\delta}) \text{ examples} \]

\[\text{Used } \mathcal{O}(\frac{1}{\epsilon^2} \ln \frac{1}{\delta}) \text{ examples} \]
Features

Clause pruning:

- Often, “bad” clauses are constructed (e.g. from noisy examples).
- These are discarded when:
  - they do not “change” (get specialized) for sufficiently enough time
  - and their score is below a threshold.

Warm-up period:

- Any-time algorithm
- In practice an output clause must have been evaluated on \( N_{\text{min}} \) examples.

Tie breaking:

- When the best & the second-best clause have very similar scores...
  - Break ties based on a pre-defined threshold, instead of waiting until the Hoeffding bound test succeeds.
Features

Clause pruning:

- Often, “bad” clauses are constructed (e.g. from noisy examples).
- These are discarded when:
  - they do not “change” (get specialized) for sufficiently enough time
  - and their score is below a threshold.

Warm-up period:

- Any-time algorithm
- In practice an output clause must have been evaluated on $N_{\text{min}}$ examples.
Features

Clause pruning:
- Often, “bad” clauses are constructed (e.g. from noisy examples).
- These are discarded when:
  - they do not “change” (get specialized) for sufficiently enough time
  - and their score is below a threshold.

Warm-up period:
- Any-time algorithm
- In practice an output clause must have been evaluated on $N_{\text{min}}$ examples.

Tie breaking:
- When the best & the second-best clause have very similar scores...
- Break ties based on a pre-defined threshold, instead of waiting until the Hoeffding bound test succeeds.
Learning Sets of Clauses

**TP**
- Annotation: holds
- Inferred: holds
- All ok!

**FP**
- Annotation: not holds
- Inferred: holds
- Incorrectly initiated by clause $R_{init}$
- Specialize $R_{init}$
- OR
- No termination clause “fires”
- Generate new termination clause

**FN**
- Annotation: holds
- Inferred: not holds
- Incorrectly terminated by clause $R_{term}$
- Specialize $R_{term}$
- OR
- No initiation clause “fires”
- Generate new initiation clause

Initiation Learner: Reward all clauses that correctly initiate the TP.
Termination Learner: Reward all clauses that correctly allow the TP to persist.
Learning Sets of Clauses

Initiation Learner

Termination Learner

Input stream

TP Annotation Inferred
holds holds

FP Annotation Inferred
not holds holds

FN Annotation Inferred
holds not holds

All ok!

Incorrectly initiated by clause $R_{init}$

No termination clause "fires"

Incorrectly terminated by clause $R_{term}$

No initiation clause "fires"

Specialize $R_{init}$

Generate new termination clause

Specialize $R_{term}$

Generate new initiation clause

Reward all clauses that correctly initiate the TP

Reward all clauses that correctly allow the TP to persist
Learning Sets of Clauses

**TP**
- Annotation: holds
- Inferred: holds

- All ok!

**FP**
- Annotation: not holds
- Inferred: holds

- Incorrectly initiated by clause $R_{init}$
  - Specialize $R_{init}$
  - Generate new termination clause

- No termination clause “fires”

**FN**
- Annotation: holds
- Inferred: not holds

- Incorrectly terminated by clause $R_{term}$
  - Specialize $R_{term}$
  - Generate new initiation clause

- No initiation clause “fires”

---

**Initiation Learner**
- Reward all clauses that correctly initiate the TP

**Termination Learner**
- Reward all clauses that correctly allow the TP to persist

---

Input stream
Learning Sets of Clauses

TP
Annotation holds
Inferred holds
All ok!

FP
Annotation not holds
Inferred holds
Incorrectly initiated by clause \( R_{init} \)
Specialize \( R_{init} \)

FP
Annotation not holds
Inferred holds
No termination clause \( "fires" \)
Generate new termination clause

FN
Annotation holds
Inferred not holds
Incorrectly terminated by clause \( R_{term} \)
Specialize \( R_{term} \)

FN
Annotation holds
Inferred not holds
No initiation clause \( "fires" \)
Generate new initiation clause

Initiation Learner
Penalize all clauses that incorrectly initiate the FP

Termination Learner
Generate new termination clause

Input stream
Learning Sets of Clauses

TP: Annotation holds, Inferred holds → All ok!
FP: Annotation not holds, Inferred holds → Incorrectly initiated by clause $R_{init}$ OR No termination clause “fires” → Specialize $R_{init}$ OR Generate new termination clause
FN: Annotation holds, Inferred not holds → Incorrectly terminated by clause $R_{term}$ OR No initiation clause “fires” → Specialize $R_{term}$ OR Generate new initiation clause

Initiation Learner: Generate new initiation clause
Termination Learner: Penalize all clauses that generate the FN

Input stream → FN
EC Axioms

\[
\text{holdsAt}(F, T + 1) \leftarrow \text{initiatedAt}(F, T).
\]

\[
\text{holdsAt}(F, T + 1) \leftarrow \text{holdsAt}(F, T), \not\text{terminatedAt}(F, T).
\]

Learnt Hypothesis $\mathcal{H}_t$:

\[
\text{initiatedAt}(\text{moving}(X, Y), T) \leftarrow \text{holdsAt}(\text{close}(X, Y, 34), T).
\]

\[
\text{terminatedAt}(\text{moving}(X, Y), T) \leftarrow \not\text{holdsAt}(\text{close}(X, Y, 34), T).
\]

Data Stream/Training Examples

Micro-batch $D_t$

\[
\text{holdsAt}(\text{moving}(id_1, id_2), 10)
\]

\[
\text{happensAt}(\text{walking}(id_1), 9),
\text{happensAt}(\text{walking}(id_2), 9),
\text{holdsAt}(\text{close}(id_1, id_2, 34), 9),
\text{holdsAt}(\text{orientation}(id_1, id_2, 45), 9)
\]

Micro-batch $D_t'$

\[
\not\text{holdsAt}(\text{moving}(id_1, id_2), 20)
\]

\[
\text{happensAt}(\text{actove}(id_1), 19),
\text{happensAt}(\text{running}(id_2), 19),
\not\text{holdsAt}(\text{close}(id_1, id_2, 34), 19),
\text{holdsAt}(\text{orientation}(id_1, id_2, 120), 19)
\]
Experimental Evaluation

- Experiments on the CAVIAR dataset for activity recognition.

- Systems compared
  - $\text{EC}_{\text{crisp}}$
    - Hand-Crafted set of rules
  - $\text{EC}_{\text{MM}}$
    - $\text{EC}_{\text{crisp}} + \text{MaxMargin}$
  - XHAIL
  - ILED
  - OLED
Experimental Evaluation

- Results from a fragment of CAVIAR

<table>
<thead>
<tr>
<th>Method</th>
<th>F$_1$-score</th>
<th>Theory size</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moving</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC$_{crisp}$</td>
<td>0.751</td>
<td>28</td>
<td>–</td>
</tr>
<tr>
<td>EC$_{MM}$</td>
<td><strong>0.890</strong></td>
<td>28</td>
<td>1692</td>
</tr>
<tr>
<td>XHAIL</td>
<td>0.841</td>
<td>14</td>
<td>7836</td>
</tr>
<tr>
<td>OLED</td>
<td>0.812</td>
<td>34</td>
<td>12</td>
</tr>
<tr>
<td>Meeting</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC$_{crisp}$</td>
<td>0.762</td>
<td>23</td>
<td>–</td>
</tr>
<tr>
<td>EC$_{MM}$</td>
<td><strong>0.863</strong></td>
<td>23</td>
<td>1133</td>
</tr>
<tr>
<td>XHAIL</td>
<td>0.861</td>
<td>15</td>
<td>7248</td>
</tr>
<tr>
<td>OLED</td>
<td>0.836</td>
<td>29</td>
<td>23</td>
</tr>
</tbody>
</table>
Experimental Evaluation

- Results from whole CAVIAR

<table>
<thead>
<tr>
<th>Method</th>
<th>F\textsubscript{1}-score</th>
<th>Theory size</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moving</td>
<td>EC\textsubscript{crisp}</td>
<td>0.677</td>
<td>28</td>
</tr>
<tr>
<td>OLED</td>
<td>0.732</td>
<td>42</td>
<td>124</td>
</tr>
<tr>
<td>EC\textsubscript{crisp}</td>
<td>0.735</td>
<td>23</td>
<td>–</td>
</tr>
<tr>
<td>Meeting</td>
<td>OLED</td>
<td>0.792</td>
<td>30</td>
</tr>
</tbody>
</table>
Experimental Evaluation

- Comparison with ILED

<table>
<thead>
<tr>
<th>Method</th>
<th>F$_1$-score</th>
<th>Theory size</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moving</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ILED</td>
<td>0.963</td>
<td>55</td>
<td>34</td>
</tr>
<tr>
<td>OLED</td>
<td>0.948</td>
<td>31</td>
<td>35</td>
</tr>
<tr>
<td>Meeting</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ILED</td>
<td>0.952</td>
<td>65</td>
<td>30</td>
</tr>
<tr>
<td>OLED</td>
<td><strong>0.953</strong></td>
<td><strong>53</strong></td>
<td><strong>42</strong></td>
</tr>
</tbody>
</table>
Conclusions & Future Work
Conclusions

Presented two scalable ILP algorithms for learning complex event definitions in the form of Event Calculus theories.

- **ILED**\(^1\)
  - Learns whole theories incrementally.
  - Full-memory approach.
  - Ensures soundness.
  - Scalability: Cost linear in the size of past experience.

- **OLED**\(^2\)
  - Online learner.
  - Learns theories by combining independent clauses.
  - Noise-tolerant.
  - Scalability:
    - Processes each example once.
    - Uses small subsets of the training data to make decisions.

---

\(^1\)https://github.com/nkatzz/ILED

\(^2\)https://github.com/nkatzz/OLED
Publications

Journals


Conferences-workshops

Future Work

- Distributed learning.
- Online learning of whole theories.
- Statistical Relational Learning.