Getting to the CORE of Complex Event Recognition

Stijn Vansummeren
UHasselt, Data Science Institute

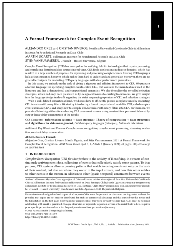
My goal for this talk

My goal for this talk

1. Present a logic for CER.
2. Introduce CEA, an automaton model for CER.
3. Explain our algorithm for processing CEA in constant-time per event.
4. Discuss limitations and open questions.

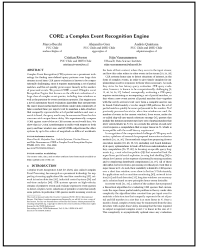
My goal for this talk

1. Present a logic for CER.
2. Introduce CEA, an automaton model for CER.
3. Explain our algorithm for processing CEA in constant-time per event.
4. Discuss limitations and open questions.



A Formal Framework for Complex Event Recognition

A Formal Framework for Complex Event Recognition
ACM TODS 46(4), 2021

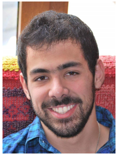


CORE: a Complex Event Recognition Engine

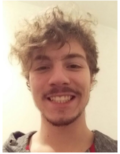
CORE: a Complex Event Recognition Engine
VLDB 2022

My goal for this talk

1. Present a logic for CER.
2. Introduce CEA, an automaton model for CER.
3. Explain our algorithm for processing CEA in constant-time per event.
4. Discuss limitations and open questions.



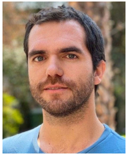
Marco Bucchi



Alejandro Grez

Andrés Quintana

Cristian Riveros



Martin Ugarte

PUC Chile, IMFD

Outline

A logic for CER

An automaton model for CER

Evaluation algorithm

The CORE complex event recognition engine

Open questions

"[...] CEP languages are often oversimplified, providing only a small set of operators, insufficient to express a number of desirable patterns and the rules to combine incoming information to produce new knowledge. Even worse, the semantics of such languages is usually given only informally, *which leads to ambiguities and makes it difficult compare the different proposals."*

G. Cugola and A. Margara

"TESLA: A formally defined event specification language", DEBS 2010.

"[...] CEP languages are often oversimplified, providing only a small set of operators, insufficient to express a number of desirable patterns and the rules to combine incoming information to produce new knowledge. Even worse, the semantics of such languages is usually given only informally*, which leads to ambiguities and makes it difficult compare the different proposals."*

G. Cugola and A. Margara

"TESLA: A formally defined event specification language", DEBS 2010.

See also [1] and [2].

[1] D. Zimmer and R. Unland
"On the semantics of complex events in active database management systems." ICDE 1999.

[2] N. Giatrakos, E. Alevizos, A. Artikis, A. Deligiannakis, M. N. Garofalakis
"Complex event recognition in the Big Data era: a survey." VLDB J. 29(1), 2020.

What do we expect for a query language for CER?

What do we expect for a query language for CER?

1. Formal syntax and semantics.

   "For every query and stream, the output will be defined precisely."

2. Declarative, denotational semantics.

   "The semantics will specify what the output is, but not how to compute it."

3. Composable language.

   "The language operators can be combined as free as possible."

What do we expect for a query language for CER?

1. Formal syntax and semantics.

   "For every query and stream, the output will be defined precisely."

2. Declarative, denotational semantics.

   "The semantics will specify what the output is, but not how to compute it."

3. Composable language.

   "The language operators can be combined as free as possible."

Complex Event Logic (CEL) is our proposal for a CER query language with these properties.

Data model for complex event recognition

Data model for complex event recognition

"A stream is a sequence of events where each event is represented as a tuple."

Event:

Buy (`vol` : 25, `stock` : 'amz')

Event type — Attribute name — Attribute value

Data model for complex event recognition

"A stream is a sequence of events where each event is represented as a tuple."

Event:

B (25, amz)

Event type

Attribute value

Data model for complex event recognition

"A stream is a sequence of events where each event is represented as a tuple."

Stream:

$\boxed{\mathrm{B}(16, a)}$ $\boxed{\mathrm{B}(23, c)}$ $\boxed{\mathrm{S}(16, b)}$ $\boxed{\mathrm{B}(25, a)}$ $\boxed{\mathrm{S}(11, c)}$ $\boxed{\mathrm{S}(12, d)}$ $\cdots$

Data model for complex event recognition

"A stream is a sequence of events where each event is represented as a tuple."

Stream:

$\mathrm{B}(16, a)$ $\mathrm{B}(23, c)$ $\mathrm{S}(16, b)$ $\mathrm{B}(25, a)$ $\mathrm{S}(11, c)$ $\mathrm{S}(12, d)$ $\cdots$

$\uparrow$

0

Data model for complex event recognition

"A stream is a sequence of events where each event is represented as a tuple."

Stream:

$\mathrm{B}(16, a)$ $\mathrm{B}(23, c)$ $\mathrm{S}(16, b)$ $\mathrm{B}(25, a)$ $\mathrm{S}(11, c)$ $\mathrm{S}(12, d)$ $\cdots$

1

Data model for complex event recognition

"A stream is a sequence of events where each event is represented as a tuple."

Stream:

$\mathrm{B}(16, a)$ $\mathrm{B}(23, c)$ $\mathrm{S}(16, b)$ $\mathrm{B}(25, a)$ $\mathrm{S}(11, c)$ $\mathrm{S}(12, d)$ $\cdots$

$\uparrow$

2

Data model for complex event recognition

"A stream is a sequence of events where each event is represented as a tuple."

Stream:

$\mathrm{B}(16, a)$ $\mathrm{B}(23, c)$ $\mathrm{S}(16, b)$ $\mathrm{B}(25, a)$ $\mathrm{S}(11, c)$ $\mathrm{S}(12, d)$ $\cdots$

3

Data model for complex event recognition

"A stream is a sequence of events where each event is represented as a tuple."

Stream:

$\mathrm{B}(16, a)$ $\mathrm{B}(23, c)$ $\mathrm{S}(16, b)$ $\mathrm{B}(25, a)$ $\mathrm{S}(11, c)$ $\mathrm{S}(12, d)$ $\cdots$

4

Data model for complex event recognition

"A stream is a sequence of events where each event is represented as a tuple."

Stream:

$\boxed{\mathrm{B}(16, a)}$ $\boxed{\mathrm{B}(23, c)}$ $\boxed{\mathrm{S}(16, b)}$ $\boxed{\mathrm{B}(25, a)}$ $\boxed{\mathrm{S}(11, c)}$ $\boxed{\mathrm{S}(12, d)}$ $\cdots$

5

Data model for complex event recognition

"A stream is a sequence of events where each event is represented as a tuple."

Stream:

$\boxed{\mathrm{B}(16, a)}$ $\boxed{\mathrm{B}(23, c)}$ $\boxed{\mathrm{S}(16, b)}$ $\boxed{\mathrm{B}(25, a)}$ $\boxed{\mathrm{S}(11, c)}$ $\boxed{\mathrm{S}(12, d)}$ $\cdots$

Data model for complex event recognition

"A stream is a sequence of events where each event is represented as a tuple."

Stream:

$\mathrm{B}_0(16, a)$ $\mathrm{B}_1(23, c)$ $\mathrm{S}_2(16, b)$ $\mathrm{B}_3(25, a)$ $\mathrm{S}_4(11, c)$ $\mathrm{S}_5(12, d)$ $\cdots$

Data model for complex event recognition

"A stream is a sequence of events where each event is represented as a tuple."

Stream:

B_0 B_1 S_2 B_3 S_4 S_5 $\cdots$

Data model for complex event recognition

Definition

A **complex event** is a pair $([i, j], C)$ where

- $[i, j]$ is an interval that denotes the start and end of the complex event;
- $C \subseteq \{i, i+1, \ldots, j\}$ is a finite set of selected events.

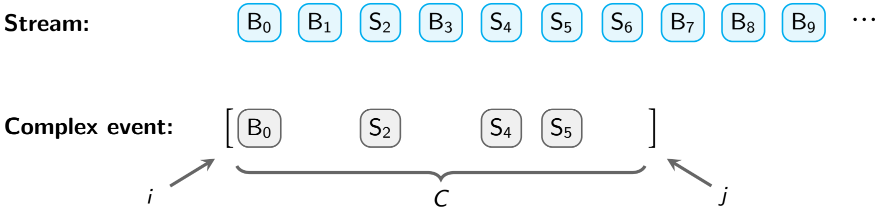
Stream: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

Data model for complex event recognition

Definition

A **complex event** is a pair $([i,j], C)$ where

- $[i,j]$ is an interval that denotes the start and end of the complex event;
- $C \subseteq \{i, i+1, \ldots, j\}$ is a finite set of selected events.



Data model for complex event recognition

Definition

A **complex event** is a pair $([i, j], C)$ where

- $[i, j]$ is an interval that denotes the start and end of the complex event;
- $C \subseteq \{i, i+1, \ldots, j\}$ is a finite set of selected events.

Stream: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

Complex event: $[\ B_0 \quad S_2 \quad S_4 \quad S_5\ ]$

Data model for complex event recognition

Definition

A **complex event** is a pair $([i, j], C)$ where

- $[i, j]$ is an interval that denotes the start and end of the complex event;
- $C \subseteq \{i, i+1, \ldots, j\}$ is a finite set of selected events.

Stream: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

Complex event: $[$ B_3 S_4 S_6 $]$

Data model for complex event recognition

" Complex Event Recognition (CER) *is the act of recognizing complex events in a stream of primitive events."*

Data model for complex event recognition

" Complex Event Recognition (CER) is the act of recognizing complex events in a stream of primitive events."

Input: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

Data model for complex event recognition

"Complex Event Recognition (CER) *is the act of recognizing complex events in a stream of primitive events.*"

Input: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

Output:

Data model for complex event recognition

> "Complex Event Recognition (CER) *is the act of recognizing complex events in a stream of primitive events.*"

Input: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

Output: $[\,B_0 \quad S_2 \quad S_4\,]$

Data model for complex event recognition

> *" Complex Event Recognition (CER) is the act of recognizing complex events in a stream of primitive events."*

Input: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

Output: $\left[B_0 \quad S_2 \quad S_4 \right]$

$\left[B_3 \quad S_4 \right]$

Data model for complex event recognition

> *"Complex Event Recognition (CER) is the act of recognizing complex events in a stream of primitive events."*

Input: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

Output: $[B_0 \; S_2 \; S_4]$

$[B_3 \; S_4]$

$[B_1 \; S_2 \; S_5 \; S_6]$

$\cdots$

Data model for complex event recognition

"Complex Event Recognition (CER) *is the act of recognizing complex events in a stream of primitive events.*"

Input: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

Output:

$$\left\{ \begin{array}{c} [\,B_0 \;\; S_2 \;\; S_4\,] \\ [\,B_3 \;\; S_4\,] \\ [\,B_1 \;\; S_2 \;\; S_5 \;\; S_6\,] \\ \cdots \end{array} \right\}$$

CER queries recognize **complex events** and extract them

Complex event logic (CEL)

Complex event logic (CEL)

CEL syntax

$$\varphi \quad := \quad R$$

- R is an event type.

Complex event logic (CEL)

CEL syntax

$$\varphi \quad := \quad R \;\mid\; \varphi\,;\,\varphi \;\mid\; \varphi \;\texttt{OR}\; \varphi \;\mid\; \varphi+$$

- R is an event type.

Complex event logic (CEL)

CEL syntax

$$\varphi \quad := \quad R \;\mid\; \varphi \,;\, \varphi \;\mid\; \varphi \;\texttt{OR}\; \varphi \;\mid\; \varphi + \;\mid\; \varphi \;\texttt{AS}\; X$$

- R is an event type.
- X is a variable.

Complex event logic (CEL)

CEL syntax

$$\varphi \quad := \quad R \;\mid\; \varphi \,;\, \varphi \;\mid\; \varphi \;\texttt{OR}\; \varphi \;\mid\; \varphi+ \;\mid\; \varphi \;\texttt{AS}\; X \;\mid\; \varphi \;\texttt{FILTER}\; P(\overline{X})$$

- R is an event type.
- X is a variable.
- $P(\overline{X})$ is a predicate over variables $\overline{X} = X_1, \ldots, X_k$.

Complex event logic (CEL)

CEL syntax

$$\varphi \;:=\; R \;\mid\; \varphi\,;\,\varphi \;\mid\; \varphi \;\texttt{OR}\; \varphi \;\mid\; \varphi+ \;\mid\; \varphi \;\texttt{AS}\; X \;\mid\; \varphi \;\texttt{FILTER}\; P(\overline{X}) \;\mid\; \pi_{\overline{X}}(\varphi)$$

- R is an event type.
- X is a variable.
- $P(\overline{X})$ is a predicate over variables $\overline{X} = X_1, \ldots, X_k$.

Complex event logic (CEL)

CEL syntax

$$\varphi \quad := \quad R \;\mid\; \varphi\,;\,\varphi \;\mid\; \varphi \;\texttt{OR}\; \varphi \;\mid\; \varphi+ \;\mid\; \varphi \;\texttt{AS}\; X \;\mid\; \varphi \;\texttt{FILTER}\; P(\overline{X}) \;\mid\; \pi_{\overline{X}}(\varphi)$$

- R is an event type.
- X is a variable.
- $P(\overline{X})$ is a predicate over variables $\overline{X} = X_1, \ldots, X_k$.

Example of a CEL formula

$$\varphi = (B\,;\,(S+ \;\texttt{AS}\; X)\,;\,B) \;\; \texttt{FILTER} \;\; \texttt{SameStock}(X)$$

Complex event logic (CEL)

CEL syntax

$$\varphi \quad := \quad R \;\mid\; \varphi \,;\, \varphi \;\mid\; \varphi \;\texttt{OR}\; \varphi \;\mid\; \varphi + \;\mid\; \varphi \;\texttt{AS}\; X \;\mid\; \varphi \;\texttt{FILTER}\; P(\overline{X}) \;\mid\; \pi_{\overline{X}}(\varphi)$$

- R is an event type.
- X is a variable.
- $P(\overline{X})$ is a predicate over variables $\overline{X} = X_1, \ldots, X_k$.

Example of a CEL formula

$$\varphi = (B \,;\, (S+ \;\texttt{AS}\; X) \,;\, B) \;\; \texttt{FILTER} \;\; \texttt{SameStock}(X)$$

Variables in CEL represent **sets of events** (i.e. complex events)

Complex event logic: semantics

Definition

Given a set of variables $\mathcal{X}$, a **valuation** V is a pair $([i,j],\mu)$ with $\mu : \mathcal{X} \to 2^{\mathbb{N}}$ a function that maps each variable $X \in \mathcal{X}$ to a finite set $\mu(X) \subseteq \{i,\ldots,j\}$.

Complex event logic: semantics

Definition

Given a set of variables $\mathcal{X}$, a **valuation** V is a pair $([i,j],\mu)$ with $\mu : \mathcal{X} \to 2^{\mathbb{N}}$ a function that maps each variable $X \in \mathcal{X}$ to a finite set $\mu(X) \subseteq \{i,\ldots,j\}$.

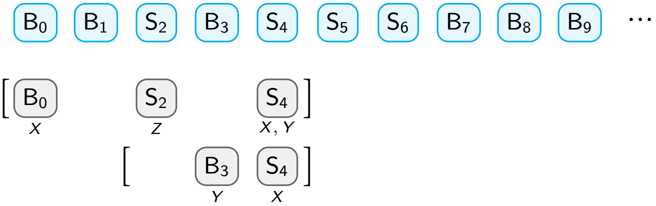
$$\boxed{B_0} \; \boxed{B_1} \; \boxed{S_2} \; \boxed{B_3} \; \boxed{S_4} \; \boxed{S_5} \; \boxed{S_6} \; \boxed{B_7} \; \boxed{B_8} \; \boxed{B_9} \; \cdots$$

$$\Big[\underset{X}{\boxed{B_0}} \qquad \underset{Z}{\boxed{S_2}} \qquad \underset{X,\,Y}{\boxed{S_4}}\Big]$$

Complex event logic: semantics

Definition

Given a set of variables $\mathcal{X}$, a **valuation** V is a pair $([i,j],\mu)$ with $\mu : \mathcal{X} \to 2^{\mathbb{N}}$ a function that maps each variable $X \in \mathcal{X}$ to a finite set $\mu(X) \subseteq \{i,\dots,j\}$.



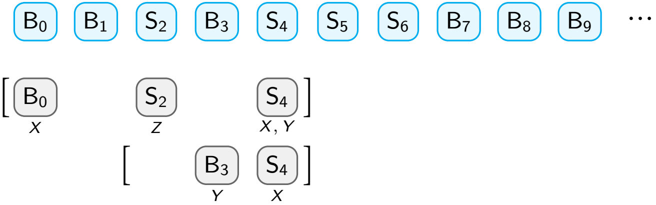
Complex event logic: semantics

Definition

Given a set of variables $\mathcal{X}$, a **valuation** V is a pair $([i,j],\mu)$ with $\mu : \mathcal{X} \to 2^{\mathbb{N}}$ a function that maps each variable $X \in \mathcal{X}$ to a finite set $\mu(X) \subseteq \{i,\ldots,j\}$.

Input:

Output:



CEL auxiliary semantics (informally)

The **valuation semantics** of CEL formula φ is a function $[\![\varphi]\!]$ that maps a **stream** $\mathcal{S}$ to a set of **valuations**.

Complex event logic: semantics

Definition

Given a set of variables $\mathcal{X}$, a **valuation** V is a pair $([i,j],\mu)$ with $\mu : \mathcal{X} \to 2^{\mathbb{N}}$ a function that maps each variable $X \in \mathcal{X}$ to a finite set $\mu(X) \subseteq \{i,\ldots,j\}$.

Input: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

Output:

$\left[\underset{X}{B_0} \quad \underset{Z}{S_2} \quad \underset{X,Y}{S_4} \right]$

$\left[\underset{Y}{B_3} \quad \underset{X}{S_4} \right]$

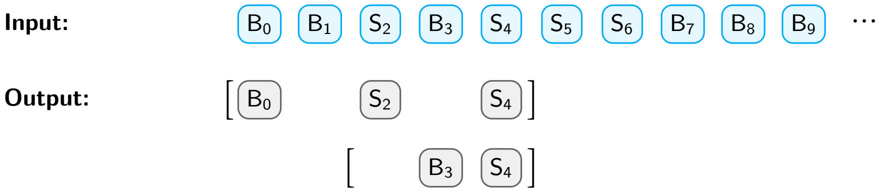
CEL semantics

The **complex event** semantics $[\![\varphi]\!]$ of CEL formula φ is obtained from $\llbracket\varphi\rrbracket$ by returning all events in the image of μ.

Complex event logic: semantics

Definition

Given a set of variables $\mathcal{X}$, a **valuation** V is a pair $([i,j],\mu)$ with $\mu : \mathcal{X} \to 2^{\mathbb{N}}$ a function that maps each variable $X \in \mathcal{X}$ to a finite set $\mu(X) \subseteq \{i,\ldots,j\}$.



CEL semantics

The **complex event** semantics $[\![\varphi]\!]$ of CEL formula φ is obtained from $\llbracket\varphi\rrbracket$ by returning all events in the image of μ.

Complex event logic: semantics

$\boxed{R}$ $\quad \varphi \,;\, \varphi \quad \varphi \ \texttt{OR} \ \varphi \quad \varphi+ \quad \varphi \ \texttt{AS} \ X \quad \varphi \ \texttt{FILTER} \ P(\overline{X}) \quad \pi_{\overline{X}}(\varphi)$

Complex event logic: semantics

R $\qquad \varphi \,;\, \varphi \qquad \varphi \texttt{ OR } \varphi \qquad \varphi+ \qquad \varphi \texttt{ AS } X \qquad \varphi \texttt{ FILTER } P(\overline{X}) \qquad \pi_{\overline{X}}(\varphi)$

$$\begin{aligned}[\![R]\!](\mathcal{S}) \quad = \; \{V \mid\; & V(\text{time}) = [i,i] \;\wedge\; \text{type}(\mathcal{S}[i]) = R \\ & \wedge\; V(R) = \{i\} \;\wedge\; \forall X \neq R.\; V(X) = \varnothing\;\}\end{aligned}$$

Complex event logic: semantics

$\boxed{R}$ $\quad \varphi ; \varphi \quad \varphi \texttt{ OR } \varphi \quad \varphi+ \quad \varphi \texttt{ AS } X \quad \varphi \texttt{ FILTER } P(\overline{X}) \quad \pi_{\overline{X}}(\varphi)$

$$
\begin{aligned}
[\![R]\!](\mathcal{S}) \quad = \{V \mid\ & V(\text{time}) = [i,i] \ \wedge\ \text{type}(\mathcal{S}[i]) = R \\
& \wedge\ V(R) = \{i\} \ \wedge\ \forall X \neq R.\ V(X) = \varnothing\ \}
\end{aligned}
$$

Example: $\varphi = B$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

$[\![\varphi]\!](\mathcal{S})$:

Complex event logic: semantics

R $\qquad \varphi \,;\, \varphi$ $\qquad \varphi$ `OR` φ $\qquad \varphi+$ $\qquad \varphi$ `AS` X $\qquad \varphi$ `FILTER` $P(\overline{X})$ $\qquad \pi_{\overline{X}}(\varphi)$

$$\begin{aligned} [\![R]\!](\mathcal{S}) \quad = \quad \{V \mid\ & V(\text{time}) = [i, i] \ \wedge\ \text{type}(\mathcal{S}[i]) = R \\ \wedge\ & V(R) = \{i\} \ \wedge\ \forall X \neq R.\ V(X) = \varnothing \ \} \end{aligned}$$

Example: $\varphi = B$

$\mathcal{S}$: $\quad B_0 \quad B_1 \quad S_2 \quad B_3 \quad S_4 \quad S_5 \quad S_6 \quad B_7 \quad B_8 \quad B_9 \quad \cdots$

$[\![\varphi]\!](\mathcal{S})$: $\quad \big[\, \underset{B}{B_0} \,\big]$

Complex event logic: semantics

R $\quad \varphi\,;\,\varphi \quad \varphi\ \texttt{OR}\ \varphi \quad \varphi+ \quad \varphi\ \texttt{AS}\ X \quad \varphi\ \texttt{FILTER}\ P(\overline{X}) \quad \pi_{\overline{X}}(\varphi)$

$$\begin{aligned}\llbracket R \rrbracket(\mathcal{S}) = \{ V \mid {} & V(\text{time}) = [i,i] \ \wedge\ \text{type}(\mathcal{S}[i]) = R \\ & \wedge\ V(R) = \{i\} \ \wedge\ \forall X \neq R.\ V(X) = \varnothing \ \}\end{aligned}$$

Example: $\varphi = B$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

$\llbracket \varphi \rrbracket(\mathcal{S})$: $\left[\underset{B}{B_0}\right]$

$\left[\underset{B}{B_1}\right]$

Complex event logic: semantics

R $\varphi\,;\,\varphi$ φ OR φ $\varphi+$ φ AS X φ FILTER $P(\overline{X})$ $\pi_{\overline{X}}(\varphi)$

$$\llbracket R \rrbracket(\mathcal{S}) = \{ V \mid V(\text{time}) = [i, i] \wedge \text{type}(\mathcal{S}[i]) = R \wedge V(R) = \{i\} \wedge \forall X \neq R.\ V(X) = \varnothing \}$$

Example: $\varphi = B$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

$\llbracket \varphi \rrbracket(\mathcal{S})$: $\Big[\underbrace{\boxed{B_0}}_{B}\Big]$ $\Big[\underbrace{\boxed{B_1}}_{B}\Big]$ $\Big[\underbrace{\boxed{B_3}}_{B}\Big]$

Complex event logic: semantics

R $\boxed{\varphi \,;\, \varphi}$ φ $\texttt{OR}$ φ $\varphi+$ φ $\texttt{AS}$ X φ $\texttt{FILTER}$ $P(\overline{X})$ $\pi_{\overline{X}}(\varphi)$

Complex event logic: semantics

R $\quad$ $\boxed{\varphi \,;\, \varphi}$ $\quad$ $\varphi\ \texttt{OR}\ \varphi$ $\quad$ $\varphi+$ $\quad$ $\varphi\ \texttt{AS}\ X$ $\quad$ $\varphi\ \texttt{FILTER}\ P(\overline{X})$ $\quad$ $\pi_{\overline{X}}(\varphi)$

$$[\![\varphi_1 \,;\, \varphi_2]\!](\mathcal{S}) \quad = \quad \{V \mid \text{there exist } V_1 \in [\![\varphi_1]\!](\mathcal{S}), V_2 \in [\![\varphi_2]\!](\mathcal{S})$$

Complex event logic: semantics

R $\quad$ $\boxed{\varphi \,;\, \varphi}$ $\quad$ $\varphi \texttt{ OR } \varphi$ $\quad$ $\varphi+$ $\quad$ $\varphi \texttt{ AS } X$ $\quad$ $\varphi \texttt{ FILTER } P(\overline{X})$ $\quad$ $\pi_{\overline{X}}(\varphi)$

$$[\![\varphi_1 \,;\, \varphi_2]\!](\mathcal{S}) \quad = \quad \{V \mid \text{there exist } V_1 \in [\![\varphi_1]\!](\mathcal{S}), V_2 \in [\![\varphi_2]\!](\mathcal{S}) \text{ s.t. } V_1(\text{end}) < V_2(\text{start})$$

Complex event logic: semantics

R $\qquad$ $\boxed{\varphi \,;\, \varphi}$ $\qquad$ φ `OR` φ $\qquad$ $\varphi+$ $\qquad$ φ `AS` X $\qquad$ φ `FILTER` $P(\overline{X})$ $\qquad$ $\pi_{\overline{X}}(\varphi)$

$$\begin{aligned}[\![\varphi_1 \,;\, \varphi_2]\!](\mathcal{S}) \quad = \{V \mid \text{there exist } V_1 \in [\![\varphi_1]\!](\mathcal{S}), V_2 \in [\![\varphi_2]\!](\mathcal{S}) \text{ s.t. } V_1(\text{end}) < V_2(\text{start}) \\ \wedge \quad V(\text{time}) = [V_1(\text{start}), V_2(\text{end})]\end{aligned}$$

Complex event logic: semantics

R $\quad$ $\boxed{\varphi\,;\,\varphi}$ $\quad$ φ $\texttt{OR}$ φ $\quad$ $\varphi+$ $\quad$ φ $\texttt{AS}$ X $\quad$ φ $\texttt{FILTER}$ $P(\overline{X})$ $\quad$ $\pi_{\overline{X}}(\varphi)$

$$
\begin{aligned}
[\![\varphi_1\,;\,\varphi_2]\!](\mathcal{S}) \quad = \{V \mid\ & \text{there exist } V_1 \in [\![\varphi_1]\!](\mathcal{S}), V_2 \in [\![\varphi_2]\!](\mathcal{S}) \text{ s.t. } V_1(\text{end}) < V_2(\text{start}) \\
\wedge \quad & V(\text{time}) = [V_1(\text{start}), V_2(\text{end})] \\
\wedge \quad & \forall X.\ V(X) = V_1(X) \cup V_2(X)\ \}
\end{aligned}
$$

Complex event logic: semantics

R $\quad \varphi \,;\, \varphi$ $\quad \varphi$ `OR` φ $\quad \varphi+$ $\quad \varphi$ `AS` X $\quad \varphi$ `FILTER` $P(\overline{X})$ $\quad \pi_{\overline{X}}(\varphi)$

$$
\begin{aligned}
[\![\varphi_1 \,;\, \varphi_2]\!](\mathcal{S}) \quad = \quad \{V \mid \; & \text{there exist } V_1 \in [\![\varphi_1]\!](\mathcal{S}), V_2 \in [\![\varphi_2]\!](\mathcal{S}) \text{ s.t. } V_1(\text{end}) < V_2(\text{start}) \\
& \wedge \quad V(\text{time}) = [V_1(\text{start}), V_2(\text{end})] \\
& \wedge \quad \forall X.\; V(X) = V_1(X) \cup V_2(X) \; \}
\end{aligned}
$$

Example: $\varphi = B \,;\, S$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

$[\![\varphi]\!](\mathcal{S})$:

Complex event logic: semantics

R $\quad$ $\boxed{\varphi\,;\,\varphi}$ $\quad$ φ `OR` φ $\quad$ $\varphi+$ $\quad$ φ `AS` X $\quad$ φ `FILTER` $P(\overline{X})$ $\quad$ $\pi_{\overline{X}}(\varphi)$

$$\begin{aligned}
[\![\varphi_1\,;\,\varphi_2]\!](\mathcal{S}) \quad = \{V \mid \text{ there exist } V_1 \in [\![\varphi_1]\!](\mathcal{S}), V_2 \in [\![\varphi_2]\!](\mathcal{S}) \text{ s.t. } V_1(\text{end}) < V_2(\text{start}) \\
\wedge \quad V(\text{time}) = [V_1(\text{start}), V_2(\text{end})] \\
\wedge \quad \forall X.\ V(X) = V_1(X) \cup V_2(X) \}
\end{aligned}$$

Example: $\varphi = B\,;\,S$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

$[\![\varphi]\!](\mathcal{S})$: $[\underset{B}{B_0} \quad \underset{S}{S_2}]$

Complex event logic: semantics

R | $\varphi \,;\, \varphi$ | φ `OR` φ | $\varphi+$ | φ `AS` X | φ `FILTER` $P(\overline{X})$ | $\pi_{\overline{X}}(\varphi)$

$$
\begin{aligned}
[\![\varphi_1 \,;\, \varphi_2]\!](\mathcal{S}) \quad = \quad \{ V \mid\ & \text{there exist } V_1 \in [\![\varphi_1]\!](\mathcal{S}), V_2 \in [\![\varphi_2]\!](\mathcal{S}) \text{ s.t. } V_1(\text{end}) < V_2(\text{start}) \\
\wedge \quad & V(\text{time}) = [V_1(\text{start}), V_2(\text{end})] \\
\wedge \quad & \forall X.\ V(X) = V_1(X) \cup V_2(X) \ \}
\end{aligned}
$$

Example: $\varphi = B \,;\, S$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

$[\![\varphi]\!](\mathcal{S})$:

$[\underset{B}{B_0} \quad \underset{S}{S_2}]$

$[\underset{B}{B_0} \quad \underset{S}{S_4}]$

Complex event logic: semantics

R $\quad \varphi ; \varphi \quad \varphi\ \texttt{OR}\ \varphi \quad \varphi+ \quad \varphi\ \texttt{AS}\ X \quad \varphi\ \texttt{FILTER}\ P(\overline{X}) \quad \pi_{\overline{X}}(\varphi)$

$$\begin{aligned}
[\![\varphi_1 ; \varphi_2]\!](\mathcal{S}) \quad = \{V \mid\ & \text{there exist } V_1 \in [\![\varphi_1]\!](\mathcal{S}), V_2 \in [\![\varphi_2]\!](\mathcal{S}) \text{ s.t. } V_1(\text{end}) < V_2(\text{start}) \\
\wedge\ & V(\text{time}) = [V_1(\text{start}), V_2(\text{end})] \\
\wedge\ & \forall X.\ V(X) = V_1(X) \cup V_2(X)\ \}
\end{aligned}$$

Example: $\varphi = B ; S$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

$[\![\varphi]\!](\mathcal{S})$:

$[\underset{B}{B_0} \quad \underset{S}{S_2}]$

$[\underset{B}{B_0} \quad \underset{S}{S_4}]$

$[\underset{B}{B_1} \quad \underset{S}{S_4}]$

Complex event logic: semantics

R $\quad\quad \varphi \,;\, \varphi \quad\quad \boxed{\varphi \texttt{ OR } \varphi} \quad\quad \varphi+ \quad\quad \varphi \texttt{ AS } X \quad\quad \varphi \texttt{ FILTER } P(\overline{X}) \quad\quad \pi_{\overline{X}}(\varphi)$

Complex event logic: semantics

R $\qquad \varphi \,;\, \varphi$ $\qquad \boxed{\varphi \ \texttt{OR}\ \varphi}$ $\qquad \varphi+$ $\qquad \varphi \ \texttt{AS}\ X$ $\qquad \varphi \ \texttt{FILTER}\ P(\overline{X})$ $\qquad \pi_{\overline{X}}(\varphi)$

$$[\![\varphi_1 \ \texttt{OR}\ \varphi_2]\!](\mathcal{S}) \quad = \quad [\![\varphi_1]\!](\mathcal{S}) \ \cup \ [\![\varphi_2]\!](\mathcal{S})$$

Complex event logic: semantics

R $\quad \varphi \,;\, \varphi$ $\quad \varphi$ OR φ $\quad \boxed{\varphi+}$ $\quad \varphi$ AS X $\quad \varphi$ FILTER $P(\overline{X})$ $\quad \pi_{\overline{X}}(\varphi)$

Complex event logic: semantics

R $\varphi \,;\, \varphi$ φ OR φ $\boxed{\varphi+}$ φ AS X φ FILTER $P(\overline{X})$ $\pi_{\overline{X}}(\varphi)$

$$[\![\varphi+]\!](\mathcal{S}) = \bigcup_{k=1}^{\infty} [\![\varphi^k]\!](\mathcal{S}) \text{ where } \varphi^k = \varphi \,;\, \cdots \,;\, \varphi \;\; k\text{-times}$$

Complex event logic: semantics

R $\qquad$ $\varphi\,;\,\varphi$ $\qquad$ φ `OR` φ $\qquad$ $\boxed{\varphi+}$ $\qquad$ φ `AS` X $\qquad$ φ `FILTER` $P(\overline{X})$ $\qquad$ $\pi_{\overline{X}}(\varphi)$

$$[\![\varphi+]\!](\mathcal{S}) = \bigcup_{k=1}^{\infty} [\![\varphi^k]\!](\mathcal{S}) \quad \text{where } \varphi^k = \varphi\,;\,\cdots\,;\,\varphi \;\; k\text{-times}$$

Example: $\varphi = B\,;\,S{+}\,;\,B$

$\mathcal{S}$: $\quad B_0 \quad B_1 \quad S_2 \quad B_3 \quad S_4 \quad S_5 \quad S_6 \quad B_7 \quad B_8 \quad B_9 \quad \cdots$

$[\![\varphi]\!](\mathcal{S})$:

Complex event logic: semantics

R $\varphi\,;\,\varphi$ φ OR φ $\boxed{\varphi+}$ φ AS X φ FILTER $P(\overline{X})$ $\pi_{\overline{X}}(\varphi)$

$$[\![\varphi+]\!](\mathcal{S}) = \bigcup_{k=1}^{\infty} [\![\varphi^k]\!](\mathcal{S}) \quad \text{where } \varphi^k = \varphi\,;\,\cdots\,;\,\varphi \ k\text{-times}$$

Example: $\varphi = B\,;\,S+\,;\,B$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

$[\![\varphi]\!](\mathcal{S})$: $\left[\underset{B}{B_0}\quad \underset{S}{S_2}\ \underset{B}{B_3}\right]$

Complex event logic: semantics

R $\qquad \varphi\,;\,\varphi$ $\qquad \varphi$ OR φ $\qquad \boxed{\varphi+}$ $\qquad \varphi$ AS X $\qquad \varphi$ FILTER $P(\overline{X})$ $\qquad \pi_{\overline{X}}(\varphi)$

$$[\![\varphi+]\!](\mathcal{S}) = \bigcup_{k=1}^{\infty} [\![\varphi^k]\!](\mathcal{S}) \quad \text{where } \varphi^k = \varphi\,;\,\cdots\,;\,\varphi \;\; k\text{-times}$$

Example: $\varphi = B\,;\,S+\,;\,B$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

$[\![\varphi]\!](\mathcal{S})$:

$$\big[\, \underset{B}{B_0} \;\; \underset{S}{S_2} \;\; \underset{B}{B_3} \,\big]$$

$$\big[\, \underset{B}{B_0} \;\; \underset{S}{S_2} \;\; \underset{S}{S_4} \;\; \underset{S}{S_5} \;\; \underset{S}{S_6} \;\; \underset{B}{B_7} \,\big]$$

Complex event logic: semantics

R $\quad\varphi\,;\,\varphi$ $\quad\varphi\ \texttt{OR}\ \varphi$ $\quad\boxed{\varphi+}$ $\quad\varphi\ \texttt{AS}\ X$ $\quad\varphi\ \texttt{FILTER}\ P(\overline{X})$ $\quad\pi_{\overline{X}}(\varphi)$

$$[\![\varphi+]\!](\mathcal{S}) \quad = \bigcup_{k=1}^{\infty} [\![\varphi^k]\!](\mathcal{S}) \quad \text{where } \varphi^k = \varphi\,;\,\cdots\,;\,\varphi \ \ k\text{-times}$$

Example: $\varphi = B\,;\,S+\,;\,B$

$\mathcal{S}$: $\quad \mathrm{B}_0 \quad \mathrm{B}_1 \quad \mathrm{S}_2 \quad \mathrm{B}_3 \quad \mathrm{S}_4 \quad \mathrm{S}_5 \quad \mathrm{S}_6 \quad \mathrm{B}_7 \quad \mathrm{B}_8 \quad \mathrm{B}_9 \quad \cdots$

$[\![\varphi]\!](\mathcal{S})$:

$$\Big[\underset{B}{\mathrm{B}_0}\quad \underset{S}{\mathrm{S}_2}\quad \underset{B}{\mathrm{B}_3}\Big]$$

$$\Big[\underset{B}{\mathrm{B}_0}\quad \underset{S}{\mathrm{S}_2}\quad \underset{S}{\mathrm{S}_4}\quad \underset{S}{\mathrm{S}_5}\quad \underset{S}{\mathrm{S}_6}\quad \underset{B}{\mathrm{B}_7}\Big]$$

$$\Big[\underset{B}{\mathrm{B}_0}\quad \underset{S}{\mathrm{S}_2}\quad \underset{S}{\mathrm{S}_5}\quad \underset{B}{\mathrm{B}_7}\Big]$$

Complex event logic: semantics

R $\varphi \,;\, \varphi$ φ OR φ $\varphi+$ $\boxed{\varphi \texttt{ AS } X}$ φ FILTER $P(\overline{X})$ $\pi_{\overline{X}}(\varphi)$

Complex event logic: semantics

R $\qquad \varphi\,;\,\varphi$ $\qquad \varphi$ OR φ $\qquad \varphi+$ $\qquad \boxed{\varphi \text{ AS } X}$ $\qquad \varphi$ FILTER $P(\overline{X})$ $\qquad \pi_{\overline{X}}(\varphi)$

$$
\begin{aligned}
[\![\varphi \text{ AS } X]\!](\mathcal{S}) \quad = \{V \mid \exists\, V' \in [\![\varphi]\!](\mathcal{S}).\ & V(\text{time}) = V'(\text{time}) \\
\wedge\ & V(X) = \textstyle\bigcup_Y V'(Y) \\
\wedge\ & \forall Z \neq X.\ V(Z) = V'(Z)\ \}
\end{aligned}
$$

Complex event logic: semantics

R $\qquad \varphi\,;\,\varphi$ $\qquad \varphi$ **OR** φ $\qquad \varphi+$ $\qquad \boxed{\varphi \textbf{ AS } X}$ $\qquad \varphi$ **FILTER** $P(\overline{X})$ $\qquad \pi_{\overline{X}}(\varphi)$

$$\begin{aligned} [\![\varphi \text{ AS } X]\!](\mathcal{S}) \quad = \{V \mid \exists\, V' \in [\![\varphi]\!](\mathcal{S}).\ & V(\text{time}) = V'(\text{time}) \\ \wedge\ & V(X) = \cup_Y V'(Y) \\ \wedge\ & \forall Z \neq X.\ V(Z) = V'(Z) \ \} \end{aligned}$$

Example: $\varphi = (B\,;\,S+) \qquad ;\,B$

$\mathcal{S}$:	B_0	B_1	S_2	B_3	S_4	S_5	S_6	B_7	B_8	B_9	$\cdots$
$[\![\varphi]\!](\mathcal{S})$:	[B_0 (B)		S_2 (S)	B_3 (B)]							
	[B_0 (B)		S_2 (S)		S_4 (S)	S_5 (S)	S_6 (S)	B_7 (B)]			
	[B_0 (B)		S_2 (S)			S_5 (S)		B_7 (B)]			

Complex event logic: semantics

R $\qquad \varphi \,;\, \varphi$ $\qquad \varphi \;\texttt{OR}\; \varphi$ $\qquad \varphi+$ $\qquad \boxed{\varphi \;\texttt{AS}\; X}$ $\qquad \varphi \;\texttt{FILTER}\; P(\overline{X})$ $\qquad \pi_{\overline{X}}(\varphi)$

$$
\begin{aligned}
[\![\varphi \;\text{AS}\; X]\!](\mathcal{S}) \quad = \quad \{V \mid \exists\, V' \in [\![\varphi]\!](\mathcal{S}).\;\; & V(\text{time}) = V'(\text{time}) \\
\wedge \;\; & V(X) = \textstyle\bigcup_Y V'(Y) \\
\wedge \;\; & \forall Z \neq X.\; V(Z) = V'(Z) \;\}
\end{aligned}
$$

Example: $\varphi = (B \,;\, S+)\;\; \text{AS} \;\; X \,;\, B$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

$[\![\varphi]\!](\mathcal{S})$:

B_0 (B, X)	S_2 (S, X)	B_3 (B)					
B_0 (B, X)	S_2 (S, X)		S_4 (S, X)	S_5 (S, X)	S_6 (S, X)	B_7 (B)
B_0 (B, X)	S_2 (S, X)			S_5 (S, X)		B_7 (B)

Complex event logic: semantics

R $\varphi \, ; \, \varphi$ $\varphi \texttt{ OR } \varphi$ $\varphi+$ $\varphi \texttt{ AS } X$ $\boxed{\varphi \texttt{ FILTER } P(\overline{X})}$ $\pi_{\overline{X}}(\varphi)$

Complex event logic: semantics

R $\varphi \,;\, \varphi$ $\varphi \texttt{ OR } \varphi$ $\varphi+$ $\varphi \texttt{ AS } X$ $\varphi \texttt{ FILTER } P(\overline{X})$ $\pi_{\overline{X}}(\varphi)$

Definition

Consider universal predicates $P(X_1, \ldots X_n)$ of the form:

$$P(X_1, \ldots X_n) \;:=\; \forall t_1 \in X_1 \ldots \forall t_n \in X_n.\; P_E(t_1, \ldots, t_n)$$

where $P_E(t_1, \ldots, t_n)$ is a **first-order predicate over tuples**.

Complex event logic: semantics

R | $\varphi \,;\, \varphi$ | φ `OR` φ | $\varphi+$ | φ `AS` X | φ `FILTER` $P(\overline{X})$ | $\pi_{\overline{X}}(\varphi)$

Definition

Consider universal predicates $P(X_1, \ldots X_n)$ of the form:

$$P(X_1, \ldots X_n) \;:=\; \forall t_1 \in X_1 \;\ldots\; \forall t_n \in X_n.\; P_E(t_1, \ldots, t_n)$$

where $P_E(t_1, \ldots, t_n)$ is a **first-order predicate over tuples**.

Examples

- `Stock=a`$(X) := \forall t \in X.\; t[\text{stock}] = \text{‘a‘}$

Complex event logic: semantics

R $\varphi\,;\,\varphi$ φ OR φ $\varphi+$ φ AS X φ FILTER $P(\overline{X})$ $\pi_{\overline{X}}(\varphi)$

Definition

Consider universal predicates $P(X_1, \ldots X_n)$ of the form:

$$P(X_1, \ldots X_n) \ := \ \forall t_1 \in X_1 \ \ldots \ \forall t_n \in X_n.\ P_E(t_1, \ldots, t_n)$$

where $P_E(t_1, \ldots, t_n)$ is a **first-order predicate over tuples**.

Examples

- Stock=a$(X) := \forall t \in X.\ t[\text{stock}] = \text{`a`}$
- SameStock$(X_1, X_2) := \forall t_1 \in X_1 . \forall t_2 \in X_2.\ t_1[\text{stock}] = t_2[\text{stock}]$

Complex event logic: semantics

R $\varphi\,;\,\varphi$ φ OR φ $\varphi+$ φ AS X **φ FILTER $P(\overline{X})$** $\pi_{\overline{X}}(\varphi)$

Definition

Consider universal predicates $P(X_1, \ldots X_n)$ of the form:

$$P(X_1, \ldots X_n) \;:=\; \forall t_1 \in X_1 \,\ldots\, \forall t_n \in X_n.\; P_E(t_1, \ldots, t_n)$$

where $P_E(t_1, \ldots, t_n)$ is a **first-order predicate over tuples**.

Examples

- $\texttt{Stock=a}(X) := \forall t \in X.\; t[\text{stock}] = \text{`a`}$
- $\texttt{SameStock}(X_1, X_2) := \forall t_1 \in X_1. \forall t_2 \in X_2.\; t_1[\text{stock}] = t_2[\text{stock}]$

The definition of CEL considers **any predicate over tuples of sets of events** but we restrict to **universal predicates** to fit our purposes.

Complex event logic: semantics

R $\quad$ $\varphi \,;\, \varphi$ $\quad$ $\varphi \texttt{ OR } \varphi$ $\quad$ $\varphi+$ $\quad$ $\varphi \texttt{ AS } X$ $\quad$ $\varphi \texttt{ FILTER } P(\overline{X})$ $\quad$ $\pi_{\overline{X}}(\varphi)$

Complex event logic: semantics

R $\quad \varphi \,;\, \varphi$ $\quad \varphi \ \texttt{OR}\ \varphi$ $\quad \varphi+$ $\quad \varphi \ \texttt{AS}\ X$ $\quad \boxed{\varphi \ \texttt{FILTER}\ P(\overline{X})}$ $\quad \pi_{\overline{X}}(\varphi)$

$$[\![\varphi \ \texttt{FILTER}\ P(\overline{X})]\!](\mathcal{S}) \quad = \quad \{ V \mid V \in [\![\varphi]\!](\mathcal{S}) \wedge V(\overline{X}) \in P \}$$

Complex event logic: semantics

R $\quad\quad$ $\varphi\,;\,\varphi$ $\quad\quad$ φ OR φ $\quad\quad$ $\varphi+$ $\quad\quad$ φ AS X $\quad\quad$ $\boxed{\varphi \texttt{ FILTER } P(\overline{X})}$ $\quad\quad$ $\pi_{\overline{X}}(\varphi)$

$$[\![\varphi \texttt{ FILTER } P(\overline{X})]\!](\mathcal{S}) \quad = \{V \mid V \in [\![\varphi]\!](\mathcal{S}) \wedge V(\overline{X}) \in P\}$$

Example: $\varphi = ((B\,;\,S+) \text{ AS } X)\,;\,B)$

$\mathcal{S}$: $\boxed{B(a)_0}$ $\boxed{B(b)_1}$ $\boxed{S(a)_2}$ $\boxed{B(c)_3}$ $\boxed{S(c)_4}$ $\boxed{S(a)_5}$ $\boxed{S(b)_6}$ $\boxed{B(a)_7}$ $\boxed{B(b)_8}$ $\boxed{B(c)_9}$ $\cdots$

$[\![\varphi]\!](\mathcal{S})$:

Complex event logic: semantics

R $\quad$ $\varphi \,;\, \varphi$ $\quad$ $\varphi \texttt{ OR } \varphi$ $\quad$ $\varphi+$ $\quad$ $\varphi \texttt{ AS } X$ $\quad$ $\boxed{\varphi \texttt{ FILTER } P(\overline{X})}$ $\quad$ $\pi_{\overline{X}}(\varphi)$

$$[\![\varphi \texttt{ FILTER } P(\overline{X})]\!](\mathcal{S}) \quad = \; \{ V \mid V \in [\![\varphi]\!](\mathcal{S}) \wedge V(\overline{X}) \in P \}$$

Example: $\varphi = ((B \,;\, S+) \text{ AS } X) \,;\, B)$

$\mathcal{S}$:	$\mathrm{B(a)}_0$	$\mathrm{B(b)}_1$	$\mathrm{S(a)}_2$	$\mathrm{B(c)}_3$	$\mathrm{S(c)}_4$	$\mathrm{S(a)}_5$	$\mathrm{S(b)}_6$	$\mathrm{B(a)}_7$	$\mathrm{B(b)}_8$	$\mathrm{B(c)}_9$	$\cdots$
$[\![\varphi]\!](\mathcal{S})$:	[$\underset{B,X}{\mathrm{B(a)}_0}$		$\underset{S,X}{\mathrm{S(a)}_2}$	$\underset{B}{\mathrm{B(c)}_3}$]							
	[$\underset{B,X}{\mathrm{B(a)}_0}$		$\underset{S,X}{\mathrm{S(a)}_2}$		$\underset{S,X}{\mathrm{S(c)}_4}$	$\underset{S,X}{\mathrm{S(a)}_5}$	$\underset{S,X}{\mathrm{S(b)}_6}$	$\underset{B}{\mathrm{B(a)}_7}$]			
	[$\underset{B,X}{\mathrm{B(a)}_0}$		$\underset{S,X}{\mathrm{S(a)}_2}$		$\underset{S,X}{\mathrm{S(c)}_4}$			$\underset{B}{\mathrm{B(a)}_7}$]			

Complex event logic: semantics

R $\qquad$ $\varphi\,;\,\varphi$ $\qquad$ φ `OR` φ $\qquad$ $\varphi+$ $\qquad$ φ `AS` X $\qquad$ **φ `FILTER` $P(\overline{X})$** $\qquad$ $\pi_{\overline{X}}(\varphi)$

$$[\![\varphi\ \texttt{FILTER}\ P(\overline{X})]\!](\mathcal{S}) \quad = \ \{ V \mid V \in [\![\varphi]\!](\mathcal{S}) \wedge V(\overline{X}) \in P \}$$

Example: $\varphi = ((B\,;\,S+)\ \text{AS}\ X)\,;\,B)\ \texttt{FILTER Stock=a}(X)$

$\mathcal{S}$: $\quad \text{B(a)}_0 \quad \text{B(b)}_1 \quad \text{S(a)}_2 \quad \text{B(c)}_3 \quad \text{S(c)}_4 \quad \text{S(a)}_5 \quad \text{S(b)}_6 \quad \text{B(a)}_7 \quad \text{B(b)}_8 \quad \text{B(c)}_9 \quad \cdots$

$[\![\varphi]\!](\mathcal{S})$:

$$\Big[\ \underset{B,X}{\text{B(a)}_0} \quad \underset{S,X}{\text{S(a)}_2} \quad \underset{B}{\text{B(c)}_3}\ \Big]$$

$$\cancel{\Big[\ \underset{B,X}{\text{B(a)}_0} \quad \underset{S,X}{\text{S(a)}_2} \quad \underset{S,X}{\text{S(c)}_4} \quad \underset{S,X}{\text{S(a)}_5} \quad \underset{S,X}{\text{S(b)}_6} \quad \underset{B}{\text{B(a)}_7}\ \Big]}$$

$$\cancel{\Big[\ \underset{B,X}{\text{B(a)}_0} \quad \underset{S,X}{\text{S(a)}_2} \quad \underset{S,X}{\text{S(c)}_4} \quad \underset{B}{\text{B(a)}_7}\ \Big]}$$

Complex event logic: semantics

R $\quad \varphi\,;\,\varphi$ $\quad \varphi$ $\texttt{OR}$ φ $\quad \varphi+$ $\quad \varphi$ $\texttt{AS}$ X $\quad \varphi$ $\texttt{FILTER}$ $P(\overline{X})$ $\quad \boxed{\pi_{\overline{X}}(\varphi)}$

Complex event logic: semantics

R $\qquad \varphi \,;\, \varphi$ $\qquad \varphi \ \mathtt{OR}\ \varphi$ $\qquad \varphi+$ $\qquad \varphi\ \mathtt{AS}\ X$ $\qquad \varphi\ \mathtt{FILTER}\ P(\overline{X})$ $\qquad \boxed{\pi_{\overline{X}}(\varphi)}$

$$
\begin{aligned}
[\![\, \pi_{\overline{X}}(\varphi)]\!](\mathcal{S}) \quad = \quad \{V \mid\ & \exists\, V' \in [\![\varphi]\!](\mathcal{S}).\ V(\text{time}) = V'(\text{time}) \\
& \wedge\ \forall Y \in \overline{X}.\ V(Y) = V'(Y) \\
& \wedge\ \ \forall Y \notin \overline{X}.\ V(Y) = \varnothing\}
\end{aligned}
$$

Complex event logic: semantics

R $\quad \varphi \,;\, \varphi \quad \varphi \texttt{ OR } \varphi \quad \varphi+ \quad \varphi \texttt{ AS } X \quad \varphi \texttt{ FILTER } P(\overline{X}) \quad \boxed{\pi_{\overline{X}}(\varphi)}$

$$\llbracket \pi_{\overline{X}}(\varphi) \rrbracket(\mathcal{S}) = \{V \mid \exists\, V' \in \llbracket \varphi \rrbracket(\mathcal{S}).\ V(\text{time}) = V'(\text{time}) \\ \wedge\ \forall Y \in \overline{X}.\ V(Y) = V'(Y) \\ \wedge\ \forall Y \notin \overline{X}.\ V(Y) = \varnothing\}$$

Example: $\varphi = \quad ((B\,;\,S+)\ \text{AS}\ X)\,;\,B)\ \texttt{FILTER}\ \texttt{Stock=a}(X)$

$\mathcal{S}$: B(a)$_0$ B(b)$_1$ S(a)$_2$ B(c)$_3$ S(c)$_4$ S(a)$_5$ S(b)$_6$ B(a)$_7$ B(b)$_8$ B(c)$_9$ $\cdots$

$\llbracket \varphi \rrbracket(\mathcal{S})$:

Complex event logic: semantics

R $\qquad \varphi \,;\, \varphi$ $\qquad \varphi \texttt{ OR } \varphi$ $\qquad \varphi+$ $\qquad \varphi \texttt{ AS } X$ $\qquad \varphi \texttt{ FILTER } P(\overline{X})$ $\qquad \boxed{\pi_{\overline{X}}(\varphi)}$

$$\begin{aligned}
[\![\, \pi_{\overline{X}}(\varphi)]\!](\mathcal{S}) \quad = \quad \{V \mid\ & \exists\, V' \in [\![\varphi]\!](\mathcal{S}).\ V(\text{time}) = V'(\text{time}) \\
& \wedge\ \forall Y \in \overline{X}.\ V(Y) = V'(Y) \\
& \wedge\ \forall Y \notin \overline{X}.\ V(Y) = \varnothing\}
\end{aligned}$$

Example: $\varphi = \quad ((B\,;\, S+)\ \text{AS}\ X)\,;\, B)\ \texttt{FILTER Stock=a}(X)$

$\mathcal{S}$: B(a)$_0$ B(b)$_1$ S(a)$_2$ B(c)$_3$ S(c)$_4$ S(a)$_5$ S(b)$_6$ B(a)$_7$ B(b)$_8$ B(c)$_9$ $\cdots$

$[\![\varphi]\!](\mathcal{S})$: $\Big[\underset{B,X}{\text{B(a)}_0}\quad \underset{S,X}{\text{S(a)}_2}\quad \underset{S,X}{\text{S(a)}_5}\quad \underset{B}{\text{B(a)}_7}\Big]$

Complex event logic: semantics

R $\quad \varphi\,;\,\varphi$ $\quad \varphi\ \texttt{OR}\ \varphi$ $\quad \varphi+$ $\quad \varphi\ \texttt{AS}\ X$ $\quad \varphi\ \texttt{FILTER}\ P(\overline{X})$ $\quad \boxed{\pi_{\overline{X}}(\varphi)}$

$$
\begin{aligned}
[\![\,\pi_{\overline{X}}(\varphi)]\!](\mathcal{S}) \;=\; \{V \mid\ & \exists\, V' \in [\![\varphi]\!](\mathcal{S}).\ V(\text{time}) = V'(\text{time}) \\
& \wedge\ \forall Y \in \overline{X}.\ V(Y) = V'(Y) \\
& \wedge\ \forall Y \notin \overline{X}.\ V(Y) = \varnothing\}
\end{aligned}
$$

Example: $\varphi = \pi_X\,[((B\,;\,S+)\ \text{AS}\ X)\,;\,B)\ \texttt{FILTER}\ \texttt{Stock=a}(X)]$

$\mathcal{S}$: $\boxed{\text{B(a)}_0}\ \boxed{\text{B(b)}_1}\ \boxed{\text{S(a)}_2}\ \boxed{\text{B(c)}_3}\ \boxed{\text{S(c)}_4}\ \boxed{\text{S(a)}_5}\ \boxed{\text{S(b)}_6}\ \boxed{\text{B(a)}_7}\ \boxed{\text{B(b)}_8}\ \boxed{\text{B(c)}_9}\ \cdots$

$[\![\varphi]\!](\mathcal{S})$: $\Big[\ \underset{X}{\boxed{\text{B(a)}_0}}\quad \underset{X}{\boxed{\text{S(a)}_2}}\quad \underset{X}{\boxed{\text{S(a)}_5}}\ \Big]$

Complex event logic: semantics

CEL semantics (final)

The **output** of a CEL formula φ over a stream $\mathcal{S}$ **at position** n is defined as:

$$[\![\varphi]\!]_n(\mathcal{S}) = \left\{ \left(V(\text{time}), \bigcup_X V(X) \right) \mid V \in [\![\varphi]\!](\mathcal{S}), V(\text{end}) = n \right\}$$

Complex event logic: semantics

CEL semantics (final)

The **output** of a CEL formula φ over a stream $\mathcal{S}$ **at position** n is defined as:

$$[\![\varphi]\!]_n(\mathcal{S}) = \left\{ \left(V(\text{time}), \bigcup_X V(X) \right) \mid V \in [\![\varphi]\!](\mathcal{S}), V(\text{end}) = n \right\}$$

All complex events that satisfy the formula are given as output

Selection strategies

CER systems includes operations to filter complex events:

Selection strategies

usually defined by an algorithm.

Selection strategies

CER systems includes operations to filter complex events:

Selection strategies

usually defined by an algorithm.

Example: skip-till-next-match in SASE

> *"a further relaxation is to remove the contiguity requirements:*
> *all irrelevant events will be skipped until the next relevant event is read." [1]*

[1] D. Gyllstrom, J. Agrawal, Y. Diao, and N. Immerman
"On supporting Kleene closure over event streams", ICDE 2008.

Selection strategies

CER systems includes operations to filter complex events:

Selection strategies

usually defined by an algorithm.

Example: skip-till-next-match in SASE

"a further relaxation is to remove the contiguity requirements: all irrelevant events will be skipped until the next relevant event is read." [1]

In CEL we declaratively formalize existing selection strategies, and propose new ones [2].

[1] D. Gyllstrom, J. Agrawal, Y. Diao, and N. Immerman
"On supporting Kleene closure over event streams", ICDE 2008.
[2] A. Grez, C. Riveros, M. Ugarte, and S. Vansummeren
"A Formal Framework for Complex Event Recognition", ACM TODS 46(4), 2021.

Outline

A logic for CER

An automaton model for CER

Evaluation algorithm

The CORE complex event recognition engine

Open questions

Complex event automata

Let P_1 be the set of all **unary predicates** over tuples.

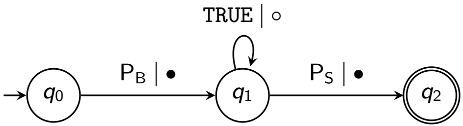
Complex event automata

Let P_1 be the set of all **unary predicates** over tuples.

Definition

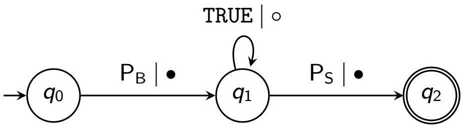
A **complex event automata** (CEA) is a tuple $\mathcal{A} = (Q, \Delta, I, F)$ where:

1. Q is a finite set of **states**,
2. I and F are the sets of **initial** and **final** states, and
3. $\Delta \subseteq Q \times \mathrm{P}_1 \times \{\bullet, \circ\} \times Q$ is the **transition relation**.



$\mathrm{P}_B \;:=\; \{\; t \;\mid\; \mathrm{type}(t) = B\}$ $\quad$ $\mathrm{P}_S \;:=\; \{\; t \;\mid\; \mathrm{type}(t) = S\}$

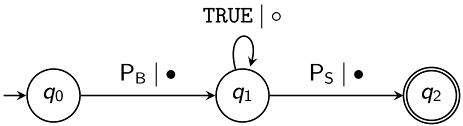
Complex event automata: semantics



$P_B := \{\, t \mid \text{type}(t) = B\}$ $\quad$ $P_S := \{\, t \mid \text{type}(t) = S\}$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

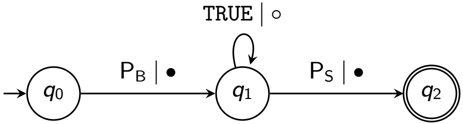
Complex event automata: semantics



$\mathrm{P_B} := \{\, t \mid \mathrm{type}(t) = B \}$ $\quad$ $\mathrm{P_S} := \{\, t \mid \mathrm{type}(t) = S \}$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

Complex event automata: semantics

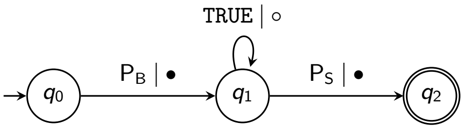


$$\mathrm{P_B} := \{\ t \mid \mathrm{type}(t) = B\} \qquad \mathrm{P_S} := \{\ t \mid \mathrm{type}(t) = S\}$$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

q_0

Complex event automata: semantics

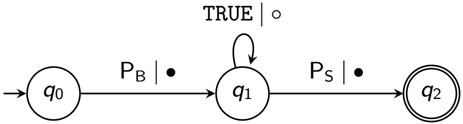


$P_B := \{\ t \mid \text{type}(t) = B\}$ $\quad$ $P_S := \{\ t \mid \text{type}(t) = S\}$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

• q_1

Complex event automata: semantics

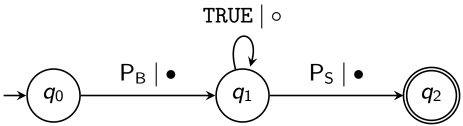


$\mathrm{P_B} := \{\ t \mid \mathrm{type}(t) = B\}$ $\quad$ $\mathrm{P_S} := \{\ t \mid \mathrm{type}(t) = S\}$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

● ○ q_1

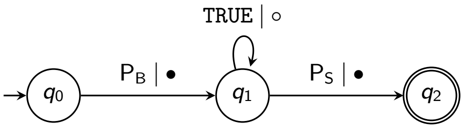
Complex event automata: semantics



$P_B := \{\, t \mid \text{type}(t) = B \}$ $\quad$ $P_S := \{\, t \mid \text{type}(t) = S \}$

$\mathcal{S}$:	B_0	B_1	S_2	B_3	S_4	S_5	S_6	B_7	B_8	B_9	$\cdots$
	•	○	• q_2								

Complex event automata: semantics

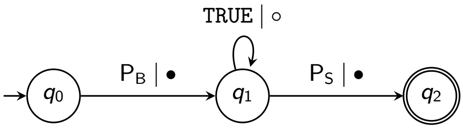


$\mathrm{P_B} := \{\, t \mid \mathrm{type}(t) = B \}$ $\quad$ $\mathrm{P_S} := \{\, t \mid \mathrm{type}(t) = S \}$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

$\bullet \quad \circ \quad \bullet$ ✓

Complex event automata: semantics



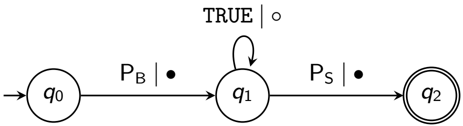
$$\mathrm{P_B} := \{\, t \mid \mathrm{type}(t) = B \}\qquad \mathrm{P_S} := \{\, t \mid \mathrm{type}(t) = S \}$$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

• ∘ • ✓

$[\![\mathcal{A}]\!](\mathcal{S})$: $[\, B_0 \quad S_2 \,]$

Complex event automata: semantics



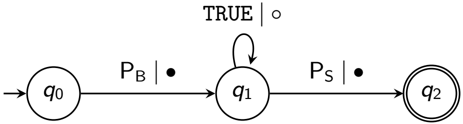
$\mathrm{P_B} := \{\, t \mid \mathrm{type}(t) = B\}$ $\quad$ $\mathrm{P_S} := \{\, t \mid \mathrm{type}(t) = S\}$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

q_0

$[\![\mathcal{A}]\!](\mathcal{S})$: $[\ \mathrm{B}_0 \quad \mathrm{S}_2\ ]$

Complex event automata: semantics



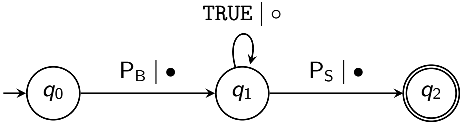
$P_B := \{ t \mid \mathrm{type}(t) = B \}$ $\quad$ $P_S := \{ t \mid \mathrm{type}(t) = S \}$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

• q_1

$[\![\mathcal{A}]\!](\mathcal{S})$: $[\, B_0 \quad S_2 \,]$

Complex event automata: semantics



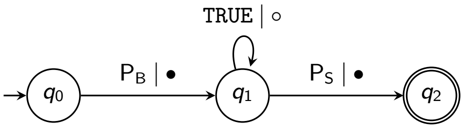
$P_B := \{\, t \mid \text{type}(t) = B \}$ $\quad$ $P_S := \{\, t \mid \text{type}(t) = S \}$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

$\bullet$ $\quad$ $\circ$ q_1

$[\![\mathcal{A}]\!](\mathcal{S})$: $\left[\; B_0 \quad S_2 \;\right]$

Complex event automata: semantics



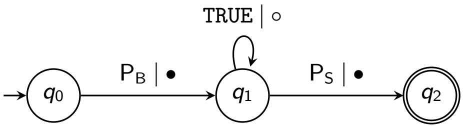
$P_B := \{ t \mid \text{type}(t) = B\}$ $\quad$ $P_S := \{ t \mid \text{type}(t) = S\}$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

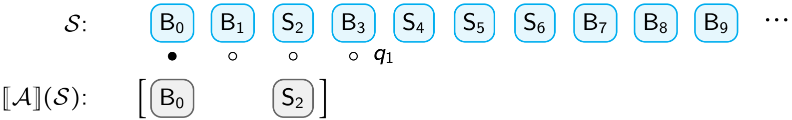
$\bullet$ $\circ$ $\circ$ q_1

$[\![\mathcal{A}]\!](\mathcal{S})$: $[\,B_0 \quad S_2\,]$

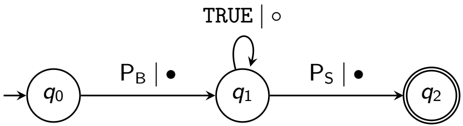
Complex event automata: semantics



$P_B := \{ t \mid \text{type}(t) = B\}$ $\quad$ $P_S := \{ t \mid \text{type}(t) = S\}$



Complex event automata: semantics



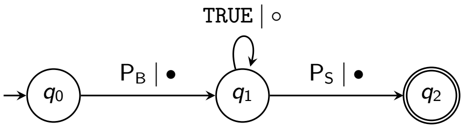
$P_B := \{ t \mid \text{type}(t) = B\}$ $\quad$ $P_S := \{ t \mid \text{type}(t) = S\}$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

$\bullet \quad \circ \quad \circ \quad \circ \quad \bullet \; q_2$

$[\![\mathcal{A}]\!](\mathcal{S})$: $[\, B_0 \quad S_2 \,]$

Complex event automata: semantics



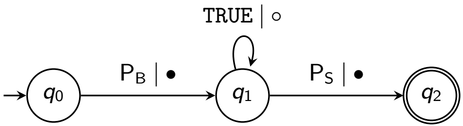
$P_B := \{\ t \mid \mathrm{type}(t) = B\}$ $\qquad$ $P_S := \{\ t \mid \mathrm{type}(t) = S\}$

$\mathcal{S}$: $B_0 \quad B_1 \quad S_2 \quad B_3 \quad S_4 \quad S_5 \quad S_6 \quad B_7 \quad B_8 \quad B_9 \quad \cdots$

$\bullet \quad \circ \quad \circ \quad \circ \quad \bullet$

$[\![\mathcal{A}]\!](\mathcal{S})$: $\left[\ B_0 \qquad S_2\ \right]$

Complex event automata: semantics



$\mathrm{P_B} := \{\, t \mid \mathrm{type}(t) = B \}$ $\quad$ $\mathrm{P_S} := \{\, t \mid \mathrm{type}(t) = S \}$

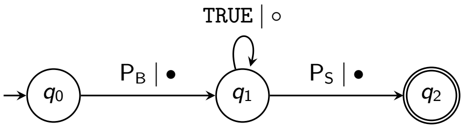
$\mathcal{S}$: $\mathrm{B_0}\ \mathrm{B_1}\ \mathrm{S_2}\ \mathrm{B_3}\ \mathrm{S_4}\ \mathrm{S_5}\ \mathrm{S_6}\ \mathrm{B_7}\ \mathrm{B_8}\ \mathrm{B_9}\ \cdots$

$\bullet\ \circ\ \circ\ \circ\ \bullet$

$[\![\mathcal{A}]\!](\mathcal{S})$: $[\,\mathrm{B_0}\quad \mathrm{S_2}\,]$

$[\,\mathrm{B_0}\quad \mathrm{S_4}\,]$

Complex event automata: semantics



$P_B := \{\, t \mid \text{type}(t) = B \}$ $\quad$ $P_S := \{\, t \mid \text{type}(t) = S \}$

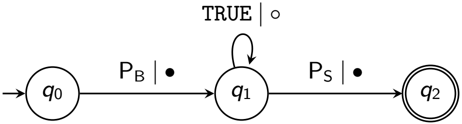
$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

q_0

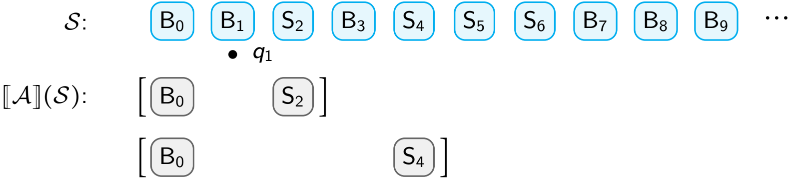
$[\![\mathcal{A}]\!](\mathcal{S})$: $[\, B_0 \quad S_2 \,]$

$[\, B_0 \quad S_4 \,]$

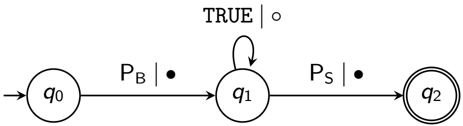
Complex event automata: semantics



$P_B := \{ t \mid \text{type}(t) = B\}$ $\quad$ $P_S := \{ t \mid \text{type}(t) = S\}$



Complex event automata: semantics



$P_B := \{ t \mid \text{type}(t) = B\}$ $\quad$ $P_S := \{ t \mid \text{type}(t) = S\}$

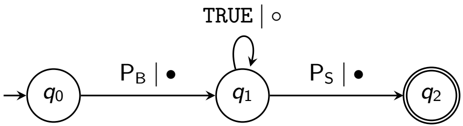
$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

• • q_2

$[\![\mathcal{A}]\!](\mathcal{S})$: $[\,B_0 \quad S_2\,]$

$[\,B_0 \quad S_4\,]$

Complex event automata: semantics



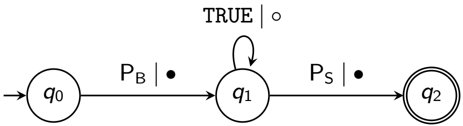
$\mathrm{P}_\mathrm{B} := \{\, t \mid \mathsf{type}(t) = B \}$ $\quad$ $\mathrm{P}_\mathrm{S} := \{\, t \mid \mathsf{type}(t) = S \}$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

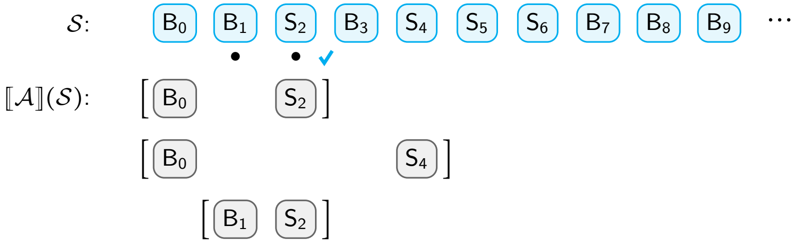
$[\![\mathcal{A}]\!](\mathcal{S})$: $[\ \mathrm{B}_0 \quad \mathrm{S}_2\ ]$

$[\ \mathrm{B}_0 \quad \mathrm{S}_4\ ]$

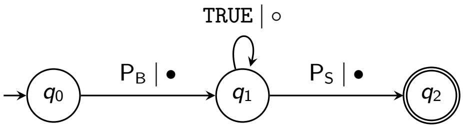
Complex event automata: semantics



$P_B := \{ t \mid \text{type}(t) = B \}$ $\quad$ $P_S := \{ t \mid \text{type}(t) = S \}$



Complex event automata: semantics



$\mathrm{P_B} := \{\, t \mid \mathrm{type}(t) = B \}$ $\quad$ $\mathrm{P_S} := \{\, t \mid \mathrm{type}(t) = S \}$

$\mathcal{S}$: $\mathrm{B_0}\ \mathrm{B_1}\ \mathrm{S_2}\ \mathrm{B_3}\ \mathrm{S_4}\ \mathrm{S_5}\ \mathrm{S_6}\ \mathrm{B_7}\ \mathrm{B_8}\ \mathrm{B_9}\ \cdots$

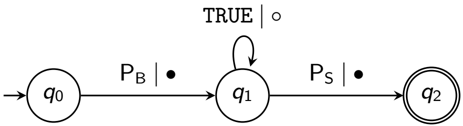
q_0

$[\![\mathcal{A}]\!](\mathcal{S})$: $[\,\mathrm{B_0}\ \mathrm{S_2}\,]$

$[\,\mathrm{B_0}\ \mathrm{S_4}\,]$

$[\,\mathrm{B_1}\ \mathrm{S_2}\,]$

Complex event automata: semantics



$P_B := \{\, t \mid \text{type}(t) = B \}$ $\quad$ $P_S := \{\, t \mid \text{type}(t) = S \}$

$\mathcal{S}$: B_0 B_1 S_2 B_3 S_4 S_5 S_6 B_7 B_8 B_9 $\cdots$

q_0

$[\![\mathcal{A}]\!](\mathcal{S})$: $[\, B_0 \quad S_2 \,]$

$[\, B_0 \quad S_4 \,]$

$[\, B_1 \quad S_2 \,]$

From CEL to CEA?

From CEL to CEA?

Theorem

For every CEL-formula φ with **unary predicate** filters we can construct a CEA $\mathcal{A}$ of size linear in φ s.t.

$$[\![\varphi]\!]_n(\mathcal{S}) = [\![\mathcal{A}]\!]_n(\mathcal{S}) \qquad \text{for every stream } \mathcal{S} \text{ and position } n.$$

From CEL to CEA?

Theorem

For every CEL-formula φ with **unary predicate** filters we can construct a CEA $\mathcal{A}$ of size linear in φ s.t.

$$[\![\varphi]\!]_n(\mathcal{S}) \;=\; [\![\mathcal{A}]\!]_n(\mathcal{S}) \qquad \text{for every stream } \mathcal{S} \text{ and position } n.$$

- CEA form a model of the "regular fragment" of CER queries.

From CEL to CEA?

Theorem

For every CEL-formula φ with **unary predicate** filters we can construct a CEA $\mathcal{A}$ of size linear in φ s.t.

$$[\![\varphi]\!]_n(\mathcal{S}) = [\![\mathcal{A}]\!]_n(\mathcal{S}) \quad \text{for every stream } \mathcal{S} \text{ and position } n.$$

- CEA form a model of the "regular fragment" of CER queries.
- **Selection strategies** can be encoded in the automaton model, see [1].

From CEL to CEA?

Theorem

For every CEL-formula φ with **unary predicate** filters we can construct a CEA $\mathcal{A}$ of size linear in φ s.t.

$$[\![\varphi]\!]_n(\mathcal{S}) = [\![\mathcal{A}]\!]_n(\mathcal{S}) \quad \text{for every stream } \mathcal{S} \text{ and position } n.$$

- CEA form a model of the "regular fragment" of CER queries.
- **Selection strategies** can be encoded in the automaton model, see [1].
- CEL can be extended to capture the expressive power of CEA, see [1].

[1] A. Grez, C. Riveros, M. Ugarte, and S. Vansummeren
"A Formal Framework for Complex Event Recognition", ACM TODS 46(4), 2021.

Outline

A logic for CER

An automaton model for CER

Evaluation algorithm

The CORE complex event recognition engine

Open questions

The partial match problem in current engines

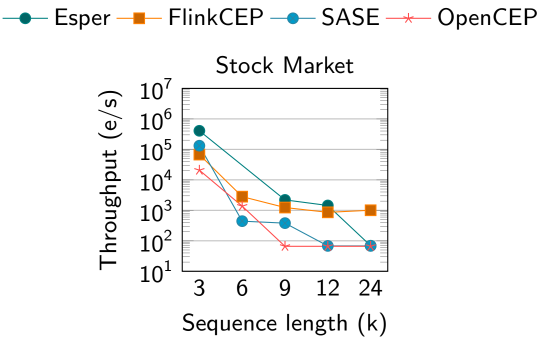
```
FROM      StockMarketStream
PATTERN   BUY b1, BUY b2, ...  , BUY bk
WITHIN    10 seconds
RETURN    b1, b2, ..., bk
```

(Written in SASE+ language)

The partial match problem in current engines

```
FROM       StockMarketStream
PATTERN    BUY b1, BUY b2, ...   , BUY bk
WITHIN     10 seconds
RETURN     b1, b2, ..., bk
```

(Written in SASE+ language)



Overcoming the partial match problem

Overcoming the partial match problem

Main idea

"Separate the streaming evaluation of CEA $\mathcal{A}$ into two processes"

Overcoming the partial match problem

Main idea

"Separate the streaming evaluation of CEA $\mathcal{A}$ into two processes"

1. **Update on each event**

Overcoming the partial match problem

Main idea

"Separate the streaming evaluation of CEA $\mathcal{A}$ into two processes"

1. **Update on each event**
   - We keep a compact representation T of partial outputs (runs).

Overcoming the partial match problem

Main idea

"Separate the streaming evaluation of CEA $\mathcal{A}$ into two processes"

1. **Update on each event**
   - We keep a compact representation T of partial outputs (runs).
   - For each new event e, we take linear time $|e| + |\mathcal{A}|$ to update T, independently of $|T|$.

Overcoming the partial match problem

Main idea

"Separate the streaming evaluation of CEA $\mathcal{A}$ into two processes"

1. **Update on each event**
   - We keep a compact representation T of partial outputs (runs).
   - For each new event e, we take linear time $|e| + |\mathcal{A}|$ to update T, independently of $|T|$.
2. **Enumeration of outputs** (output-linear delay enumeration)

Overcoming the partial match problem

Main idea

"Separate the streaming evaluation of CEA $\mathcal{A}$ into two processes"

1. **Update on each event**
   - We keep a compact representation T of partial outputs (runs).
   - For each new event e, we take linear time $|e| + |\mathcal{A}|$ to update T, independently of $|T|$.
2. **Enumeration of outputs** (output-linear delay enumeration)
   - Whenever an event triggers new recognized complex events, the enumeration phase is called, independent of the update process.

Overcoming the partial match problem

Main idea

"Separate the streaming evaluation of CEA $\mathcal{A}$ into two processes"

1. **Update on each event**
   - We keep a compact representation T of partial outputs (runs).
   - For each new event e, we take linear time $|e| + |\mathcal{A}|$ to update T, independently of $|T|$.
2. **Enumeration of outputs** (output-linear delay enumeration)
   - Whenever an event triggers new recognized complex events, the enumeration phase is called, independent of the update process.
   - All complex events $C_1, C_2, \ldots$ for the current position are enumerated taking $\mathcal{O}(|C_i|)$ time to print C_i.

Overcoming the partial match problem

Main idea

"Separate the streaming evaluation of CEA $\mathcal{A}$ into two processes"

1. **Update on each event**

   "Same guarantee as a streaming algorithm."

2. **Enumeration of outputs** (output-linear delay enumeration)
   - Whenever an event triggers new recognized complex events, the enumeration phase is called, independent of the update process.
   - All complex events $C_1, C_2, \ldots$ for the current position are enumerated taking $\mathcal{O}(|C_i|)$ time to print C_i.

Overcoming the partial match problem

Main idea

"Separate the streaming evaluation of CEA $\mathcal{A}$ into two processes"

1. **Update on each event**

   "Same guarantee as a streaming algorithm."

2. **Enumeration of outputs** (output-linear delay enumeration)

   "Users do not see any difference compared to naively storing all outputs"

Overcoming the partial match problem

Main idea

"Separate the streaming evaluation of CEA $\mathcal{A}$ into two processes"

1. **Update on each event**

   "Same guarantee as a streaming algorithm."

2. **Enumeration of outputs** (output-linear delay enumeration)

   "Users do not see any difference compared to naively storing all outputs"

If an evaluation algorithm E satisfies 1. and 2.,
we say that E has **output-linear delay** evaluation.

CEA evaluation strategy

Definition

Let $\epsilon \in \mathbb{N} \cup \{\infty\}$, let $\mathcal{A}$ be a CEA and $\mathcal{S}$ a stream. We define

$$[\![\mathcal{A} \texttt{ WITHIN } \epsilon]\!](\mathcal{S}) := \{C \in [\![\mathcal{A}]\!](\mathcal{S}) \mid C(\mathsf{end}) - C(\mathsf{start}) \leq \epsilon\}.$$

CEA evaluation strategy

Theorem

$[\![\mathcal{A}\ \texttt{WITHIN}\ \epsilon]\!]$ can be evaluated with **output-linear delay**, for every CEA $\mathcal{A}$ and every ϵ.

CEA evaluation strategy

Theorem

$[\![\mathcal{A} \ \texttt{WITHIN} \ \epsilon]\!]$ can be evaluated with **output-linear delay**, for every CEA $\mathcal{A}$ and every ϵ.

Main ideas of the algorithm:

1. A notion of **I/O deterministic** CEA.

CEA evaluation strategy

Theorem

$[\![\mathcal{A}\ \texttt{WITHIN}\ \epsilon]\!]$ can be evaluated with **output-linear delay**, for every CEA $\mathcal{A}$ and every ϵ.

Main ideas of the algorithm:

1. A notion of **I/O deterministic** CEA.
2. A **timed Enumerable Compact Set** (tECS) for compactly representing complex events and enumerating all outputs with window-size ϵ.

CEA evaluation strategy

Theorem

$[\![\mathcal{A}\ \texttt{WITHIN}\ \epsilon]\!]$ can be evaluated with **output-linear delay**, for every CEA $\mathcal{A}$ and every ϵ.

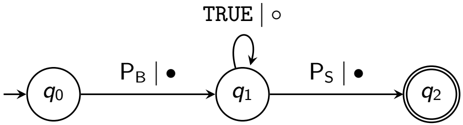
Main ideas of the algorithm:

1. A notion of **I/O deterministic** CEA.
2. A **timed Enumerable Compact Set** (tECS) for compactly representing complex events and enumerating all outputs with window-size ϵ.
3. An evaluation algorithm for incrementally building tECS given active states of I/O deterministic CEA.

I/O determinism

Definition

A CEA is **I/O deterministic** if for every pair of transitions $q \xrightarrow{P_1/m_1} q_1$ and $q \xrightarrow{P_2/m_2} q_2$ from the same state q, if $P_1 \cap P_2 \neq \emptyset$ then $m_1 \neq m_2$.

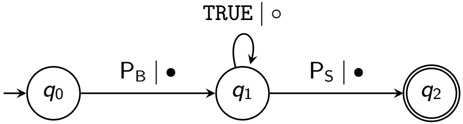


$\mathrm{P_B} := \{\, t \mid \mathrm{type}(t) = B \}$ $\quad$ $\mathrm{P_S} := \{\, t \mid \mathrm{type}(t) = S \}$

I/O determinism

Definition

A CEA is **I/O deterministic** if for every pair of transitions $q \xrightarrow{P_1/m_1} q_1$ and $q \xrightarrow{P_2/m_2} q_2$ from the same state q, if $P_1 \cap P_2 \neq \emptyset$ then $m_1 \neq m_2$.



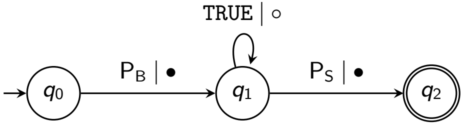
$\mathrm{P}_{\mathrm{B}} := \{\ t \mid \mathrm{type}(t) = B\}$ $\quad$ $\mathrm{P}_{\mathrm{S}} := \{\ t \mid \mathrm{type}(t) = S\}$

"Every recognized complex event has only one run that defines it."

I/O determinism

Definition

A CEA is **I/O deterministic** if for every pair of transitions $q \xrightarrow{P_1/m_1} q_1$ and $q \xrightarrow{P_2/m_2} q_2$ from the same state q, if $P_1 \cap P_2 \neq \varnothing$ then $m_1 \neq m_2$.



$\mathrm{P_B} := \{\, t \mid \mathrm{type}(t) = B \}$ $\quad$ $\mathrm{P_S} := \{\, t \mid \mathrm{type}(t) = S \}$

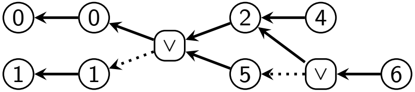
Proposition

CEA can be I/O-determinized in exponential time.

Timed Enumerable Compact Sets

Definition

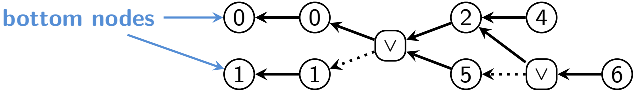
A **timed Enumerable Compact Set** (tECS) is a DAG with three kinds of nodes: **bottom** nodes, **position** nodes, and **union** nodes, with out-degree 0, 1, and 2, respectively.



Timed Enumerable Compact Sets

Definition

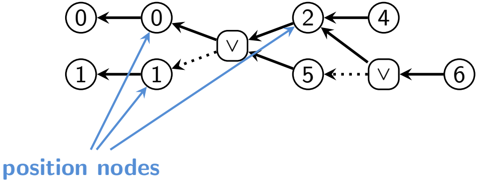
A **timed Enumerable Compact Set** (tECS) is a DAG with three kinds of nodes: **bottom** nodes, **position** nodes, and **union** nodes, with out-degree 0, 1, and 2, respectively.



Timed Enumerable Compact Sets

Definition

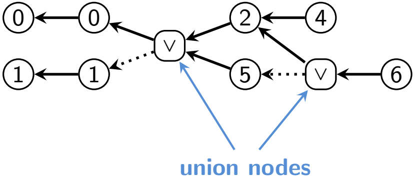
A **timed Enumerable Compact Set** (tECS) is a DAG with three kinds of nodes: **bottom** nodes, **position** nodes, and **union** nodes, with out-degree 0, 1, and 2, respectively.



Timed Enumerable Compact Sets

Definition

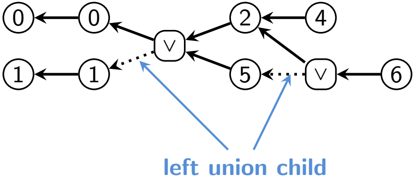
A **timed Enumerable Compact Set** (tECS) is a DAG with three kinds of nodes: **bottom** nodes, **position** nodes, and **union** nodes, with out-degree 0, 1, and 2, respectively.



Timed Enumerable Compact Sets

Definition

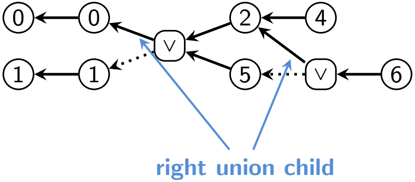
A **timed Enumerable Compact Set** (tECS) is a DAG with three kinds of nodes: **bottom** nodes, **position** nodes, and **union** nodes, with out-degree 0, 1, and 2, respectively.



Timed Enumerable Compact Sets

Definition

A **timed Enumerable Compact Set** (tECS) is a DAG with three kinds of nodes: **bottom** nodes, **position** nodes, and **union** nodes, with out-degree 0, 1, and 2, respectively.



Timed Enumerable Compact Sets: semantics

Definition

A **open complex event** is a pair (i, C) with $i \in \mathbb{N}$ and $C \subseteq \mathbb{N}$ finite.

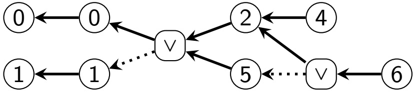
Timed Enumerable Compact Sets: semantics

Definition

A **open complex event** is a pair (i, C) with $i \in \mathbb{N}$ and $C \subseteq \mathbb{N}$ finite.

Semantics:

- Every path from a node to a bottom node defines an open complex event.
- A node n hence encodes a set $[\![n]\!]$ of open complex events.



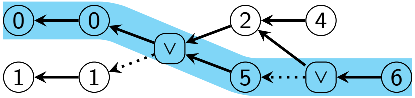
Timed Enumerable Compact Sets: semantics

Definition

A **open complex event** is a pair (i, C) with $i \in \mathbb{N}$ and $C \subseteq \mathbb{N}$ finite.

Semantics:

- Every path from a node to a bottom node defines an open complex event.
- A node n hence encodes a set $[\![n]\!]$ of open complex events.



Open complex event: $(0, \{0, 5, 6\})$

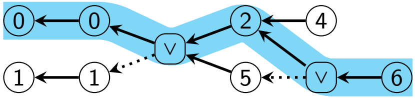
Timed Enumerable Compact Sets: semantics

Definition

A **open complex event** is a pair (i, C) with $i \in \mathbb{N}$ and $C \subseteq \mathbb{N}$ finite.

Semantics:

- Every path from a node to a bottom node defines an open complex event.
- A node n hence encodes a set $[\![n]\!]$ of open complex events.



Open complex event: $(0, \{0, 2, 6\})$

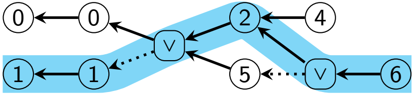
Timed Enumerable Compact Sets: semantics

Definition

A **open complex event** is a pair (i, C) with $i \in \mathbb{N}$ and $C \subseteq \mathbb{N}$ finite.

Semantics:

- Every path from a node to a bottom node defines an open complex event.
- A node n hence encodes a set $[\![n]\!]$ of open complex events.



Open complex event: $(1, \{1, 2, 6\})$

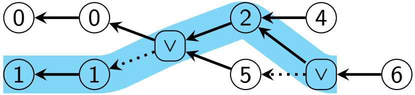
Timed Enumerable Compact Sets: semantics

Definition

A **open complex event** is a pair (i, C) with $i \in \mathbb{N}$ and $C \subseteq \mathbb{N}$ finite.

Semantics:

- Every path from a node to a bottom node defines an open complex event.
- A node n hence encodes a set $[\![n]\!]$ of open complex events.



Open complex event: $(1, \{1, 2\})$

Timed Enumerable Compact Sets: enumeration

For each position node n, window size ϵ and $j \in \mathbb{N}$ we want to be able to enumerate

$$[\![n]\!]^{\epsilon}(j) := \{ \qquad \mid (i, C) \in [\![n]\!] \qquad \}$$

with output-linear delay.

Timed Enumerable Compact Sets: enumeration

For each position node n, window size ϵ and $j \in \mathbb{N}$ we want to be able to enumerate

$$[\![n]\!]^{\epsilon}(j) := \{ \qquad\qquad \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon \}$$

with output-linear delay.

Timed Enumerable Compact Sets: enumeration

For each position node n, window size ϵ and $j \in \mathbb{N}$ we want to be able to enumerate

$$[\![n]\!]^{\epsilon}(j) := \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Timed Enumerable Compact Sets: enumeration

For each position node n, window size ϵ and $j \in \mathbb{N}$ we want to be able to enumerate

$$[\![n]\!]^{\epsilon}(j) := \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

In order to allow this, we need the following structure on tECS:

Timed Enumerable Compact Sets: enumeration

For each position node n, window size ϵ and $j \in \mathbb{N}$ we want to be able to enumerate

$$[\![n]\!]^{\epsilon}(j) := \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

In order to allow this, we need the following structure on tECS:

- For every node n, distinct paths starting at n encode distinct open complex events.

Timed Enumerable Compact Sets: enumeration

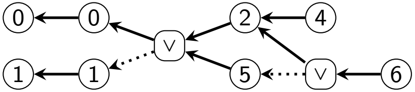
For each position node n, window size ϵ and $j \in \mathbb{N}$ we want to be able to enumerate

$$[\![n]\!]^{\epsilon}(j) := \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

In order to allow this, we need the following structure on tECS:

- For every node n, distinct paths starting at n encode distinct open complex events.



Timed Enumerable Compact Sets: enumeration

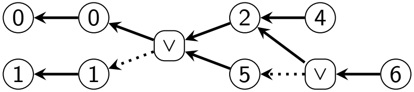
For each position node n, window size ϵ and $j \in \mathbb{N}$ we want to be able to enumerate

$$[\![n]\!]^{\epsilon}(j) := \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

In order to allow this, we need the following structure on tECS:

- For every node n, distinct paths starting at n encode distinct open complex events.
- Nodes store their **max-start** time: the largest time value of any bottom node reachable from n.



Timed Enumerable Compact Sets: enumeration

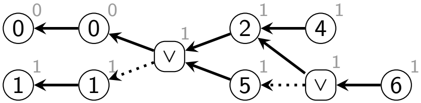
For each position node n, window size ϵ and $j \in \mathbb{N}$ we want to be able to enumerate

$$[\![n]\!]^{\epsilon}(j) := \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

In order to allow this, we need the following structure on tECS:

- For every node n, distinct paths starting at n encode distinct open complex events.
- Nodes store their **max-start** time: the largest time value of any bottom node reachable from n.



Timed Enumerable Compact Sets: enumeration

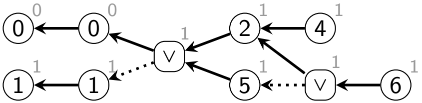
For each position node n, window size ϵ and $j \in \mathbb{N}$ we want to be able to enumerate

$$[\![n]\!]^{\epsilon}(j) := \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

In order to allow this, we need the following structure on tECS:

- For every node n, distinct paths starting at n encode distinct open complex events.
- Nodes store their **max-start** time: the largest time value of any bottom node reachable from n.
- The children of union nodes u are max-start sorted: $\max(\text{left}(u)) \geq \max(\text{right}(u))$.



Timed Enumerable Compact Sets: enumeration

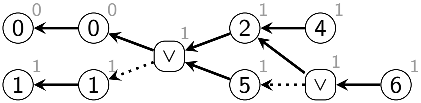
For each position node n, window size ϵ and $j \in \mathbb{N}$ we want to be able to enumerate

$$[\![n]\!]^{\epsilon}(j) := \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

In order to allow this, we need the following structure on tECS:

- For every node n, distinct paths starting at n encode distinct open complex events.
- Nodes store their **max-start** time: the largest time value of any bottom node reachable from n.
- The children of union nodes u are max-start sorted: $\max(\text{left}(u)) \geq \max(\text{right}(u))$.
- There is a constant bounding the length of chains of union left-child paths.



Timed Enumerable Compact Sets: enumeration

Theorem

Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Timed Enumerable Compact Sets: enumeration

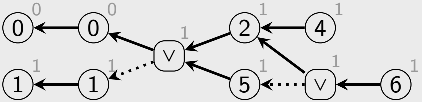
Theorem

Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6$, $\epsilon = 5$, $j = 6$



Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Timed Enumerable Compact Sets: enumeration

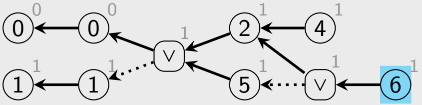
Theorem

Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6$, $\epsilon = 5$, $j = 6$



Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Timed Enumerable Compact Sets: enumeration

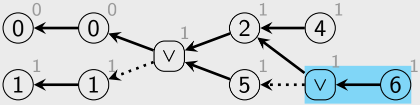
Theorem

Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j],C) \mid (i,C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6$, $\epsilon = 5$, $j = 6$



Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Timed Enumerable Compact Sets: enumeration

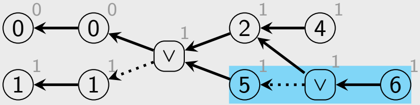
Theorem

Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6,\ \epsilon = 5,\ j = 6$



Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Timed Enumerable Compact Sets: enumeration

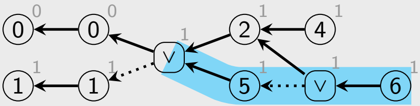
Theorem

Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6,\ \epsilon = 5,\ j = 6$



Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Timed Enumerable Compact Sets: enumeration

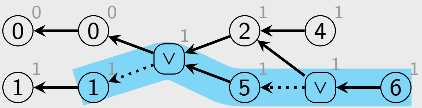
Theorem

Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6$, $\epsilon = 5$, $j = 6$



Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Timed Enumerable Compact Sets: enumeration

Theorem

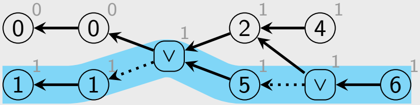
Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j],C) \mid (i,C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6$, $\epsilon = 5$, $j = 6$

output $([1,6],\{1,5,6\})$



Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Timed Enumerable Compact Sets: enumeration

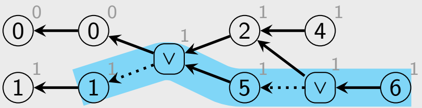
Theorem

Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6$, $\epsilon = 5$, $j = 6$



Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Timed Enumerable Compact Sets: enumeration

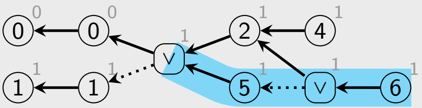
Theorem

Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6$, $\epsilon = 5$, $j = 6$



Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Timed Enumerable Compact Sets: enumeration

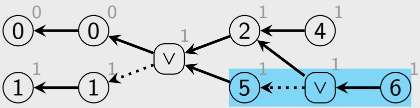
Theorem

Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6$, $\epsilon = 5$, $j = 6$



Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Timed Enumerable Compact Sets: enumeration

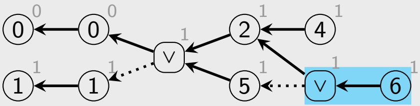
Theorem

Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6$, $\epsilon = 5$, $j = 6$



Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Timed Enumerable Compact Sets: enumeration

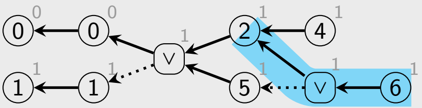
Theorem

Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6$, $\epsilon = 5$, $j = 6$



Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Timed Enumerable Compact Sets: enumeration

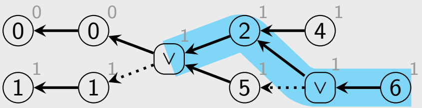
Theorem

Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j],C) \mid (i,C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6$, $\epsilon = 5$, $j = 6$



Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Timed Enumerable Compact Sets: enumeration

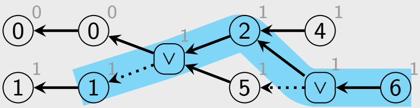
Theorem

Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6$, $\epsilon = 5$, $j = 6$



Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Timed Enumerable Compact Sets: enumeration

Theorem

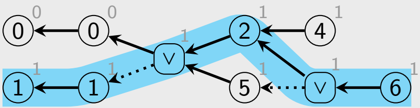
Under the previous conditions, we may enumerate

$$[\![n]\!]^{\epsilon}(j) = \{([i,j], C) \mid (i, C) \in [\![n]\!],\ j - i \leq \epsilon\}$$

with output-linear delay.

Example: $n = 6,\ \epsilon = 5,\ j = 6$

output $([1,6], \{1,2,6\})$



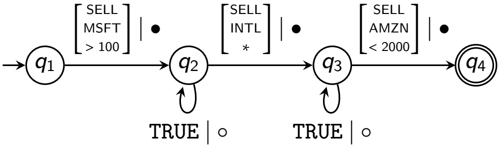
Enumeration algorithm:

- Do depth-first search, starting from n.
- Visit left-children of union nodes before right-children.
- Before moving to a child c, check that $j - \max(c) \leq \epsilon$.

Evaluation Algorithm by Example

Evaluation Algorithm by Example

CEA:

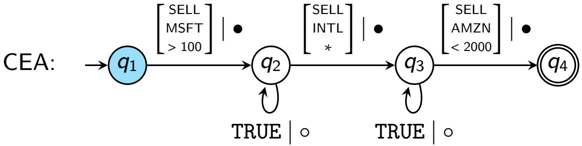


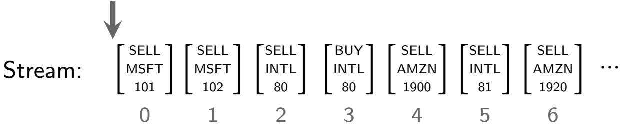
Stream:

$$\underset{0}{\begin{bmatrix}\text{SELL}\\ \text{MSFT}\\ 101\end{bmatrix}} \underset{1}{\begin{bmatrix}\text{SELL}\\ \text{MSFT}\\ 102\end{bmatrix}} \underset{2}{\begin{bmatrix}\text{SELL}\\ \text{INTL}\\ 80\end{bmatrix}} \underset{3}{\begin{bmatrix}\text{BUY}\\ \text{INTL}\\ 80\end{bmatrix}} \underset{4}{\begin{bmatrix}\text{SELL}\\ \text{AMZN}\\ 1900\end{bmatrix}} \underset{5}{\begin{bmatrix}\text{SELL}\\ \text{INTL}\\ 81\end{bmatrix}} \underset{6}{\begin{bmatrix}\text{SELL}\\ \text{AMZN}\\ 1920\end{bmatrix}} \cdots$$

Evaluation Algorithm by Example

tECS:

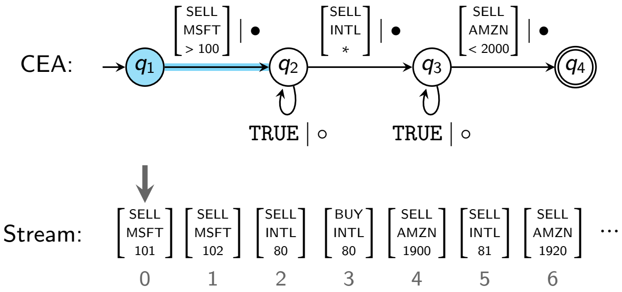




Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.

Evaluation Algorithm by Example

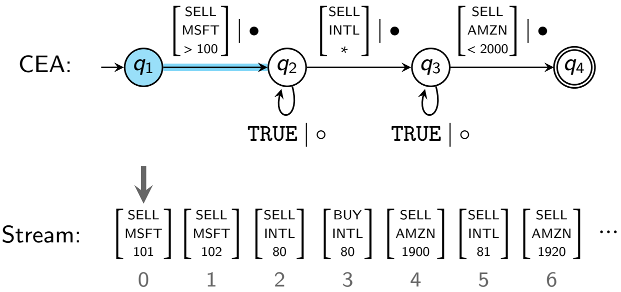


tECS:

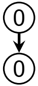
Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.

Evaluation Algorithm by Example



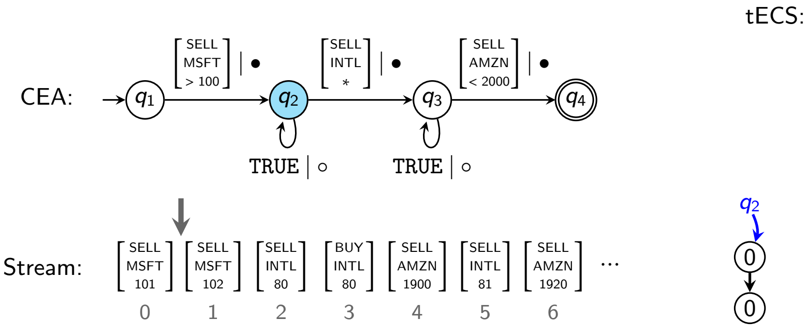
tECS:



Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.

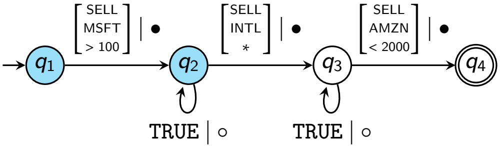
Evaluation Algorithm by Example

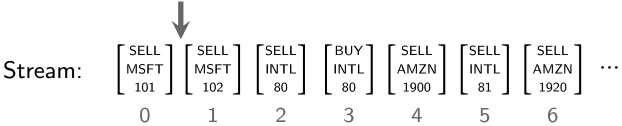


Algorithm crux:

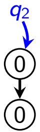
- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.

Evaluation Algorithm by Example





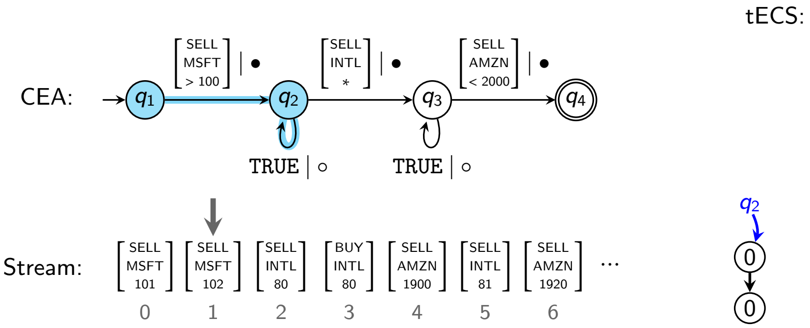
tECS:



Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.

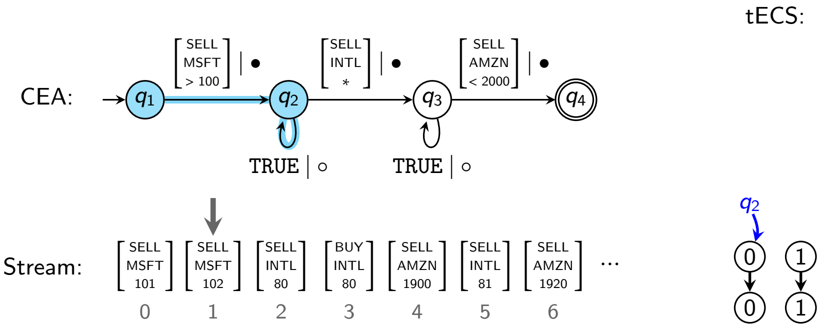
Evaluation Algorithm by Example



Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.

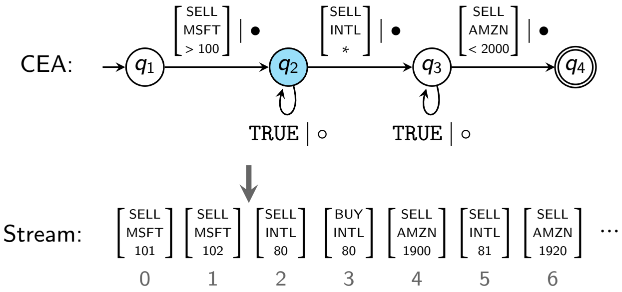
Evaluation Algorithm by Example



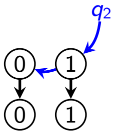
Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.

Evaluation Algorithm by Example



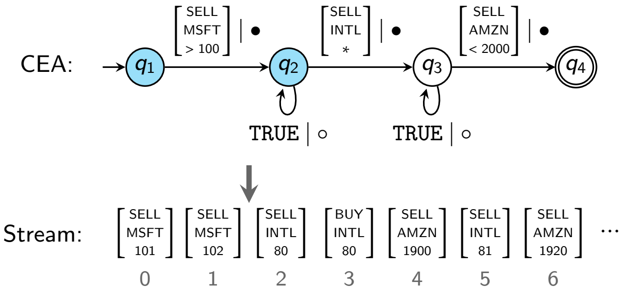
tECS:



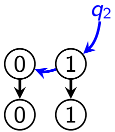
Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.

Evaluation Algorithm by Example



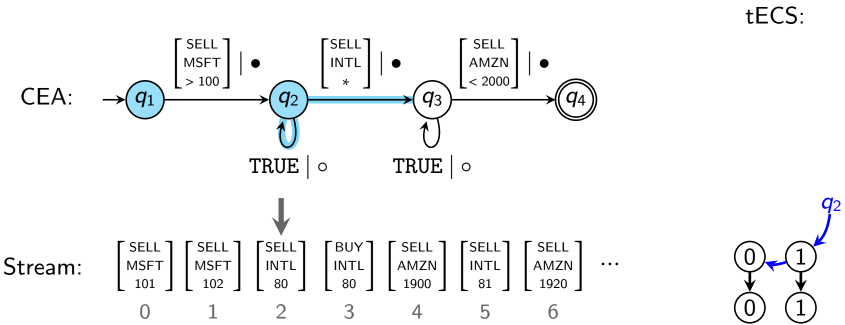
tECS:



Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.

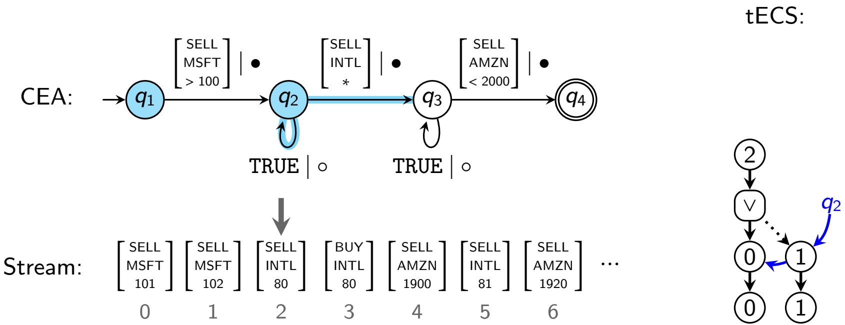
Evaluation Algorithm by Example



Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.

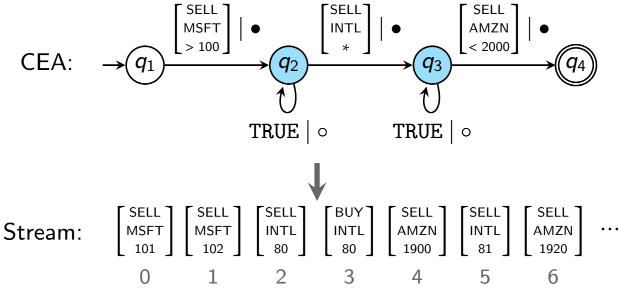
Evaluation Algorithm by Example



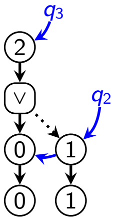
Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

Evaluation Algorithm by Example



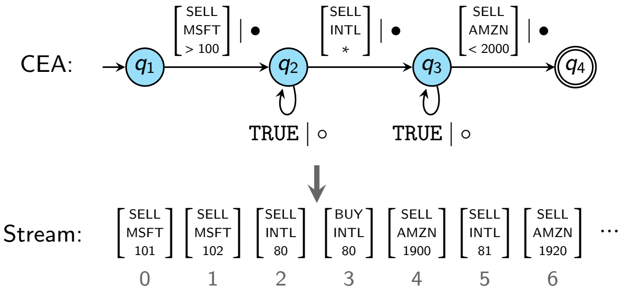
tECS:



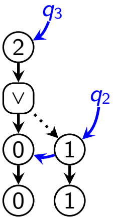
Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

Evaluation Algorithm by Example



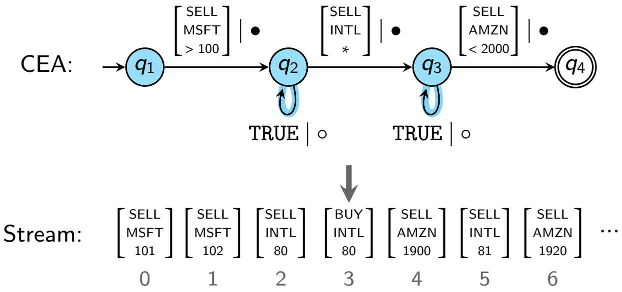
tECS:



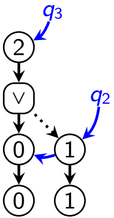
Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

Evaluation Algorithm by Example



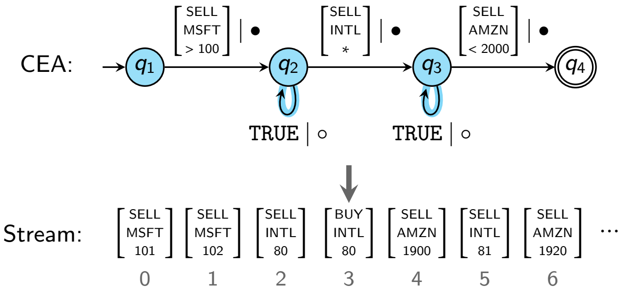
tECS:



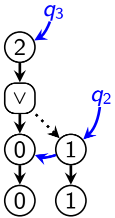
Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

Evaluation Algorithm by Example



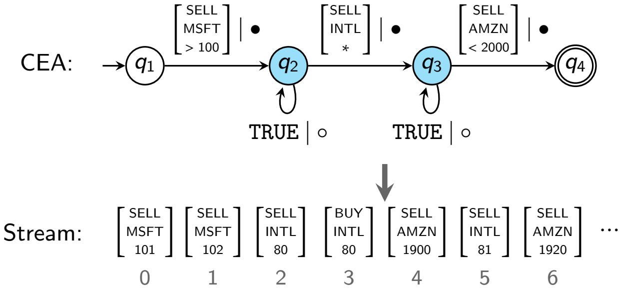
tECS:



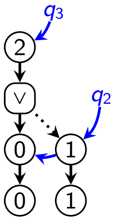
Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

Evaluation Algorithm by Example



tECS:



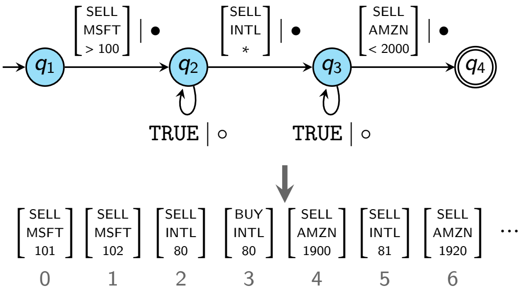
Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

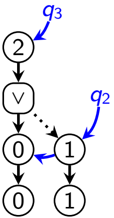
Evaluation Algorithm by Example

CEA:

Stream:



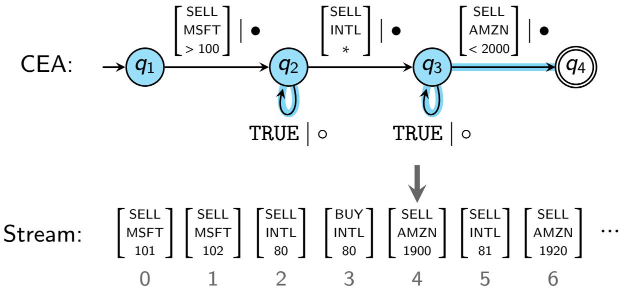
tECS:

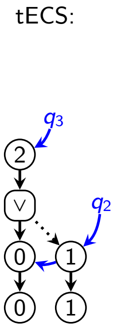


Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

Evaluation Algorithm by Example

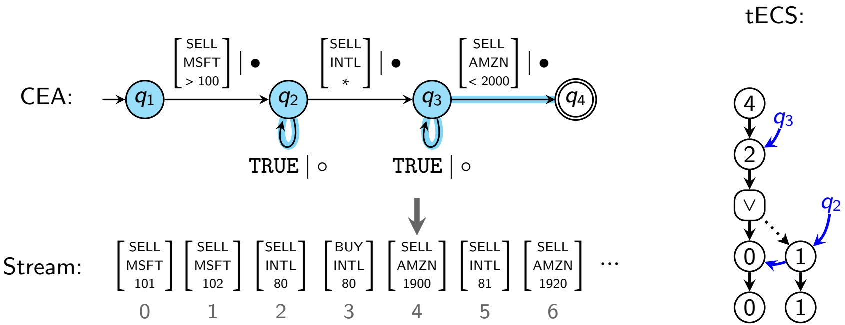




Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

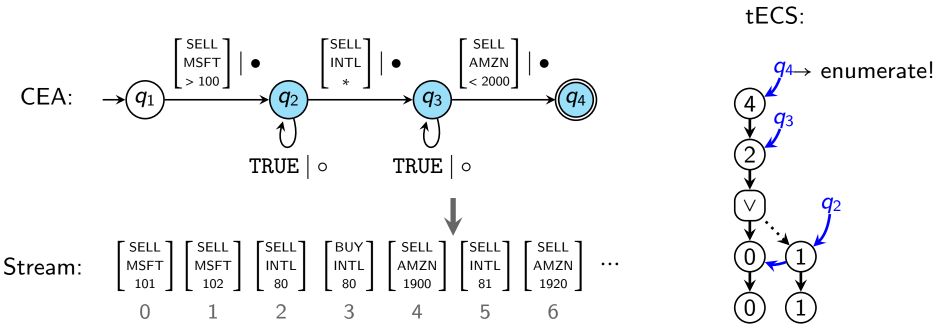
Evaluation Algorithm by Example



Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

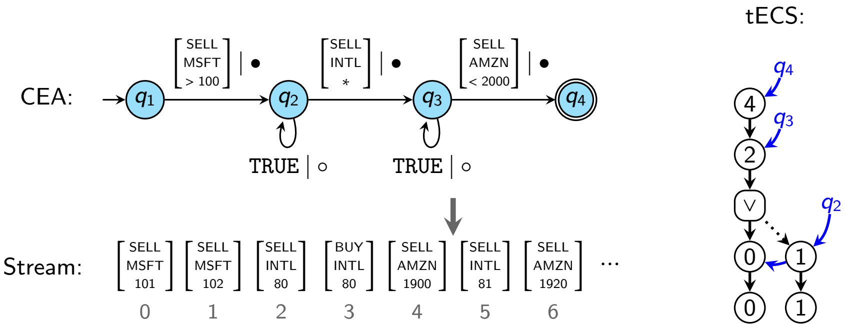
Evaluation Algorithm by Example



Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

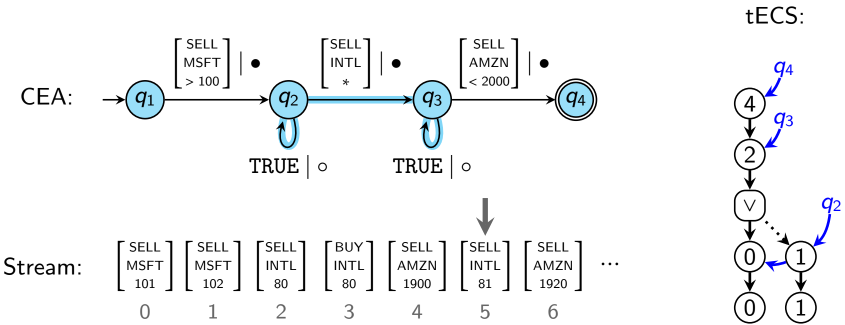
Evaluation Algorithm by Example



Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\mathrm{CEA}|)$, implying that we only take time $\mathcal{O}(|\mathrm{CEA}|)$ per event. This is constant in data complexity.

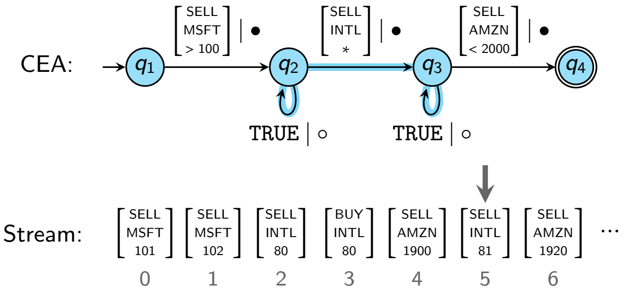
Evaluation Algorithm by Example

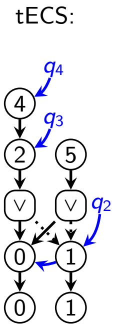


Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

Evaluation Algorithm by Example

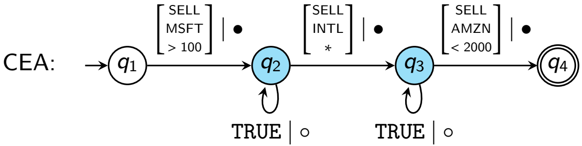


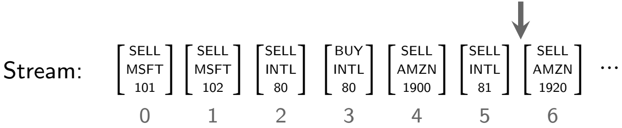


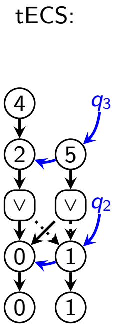
Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

Evaluation Algorithm by Example



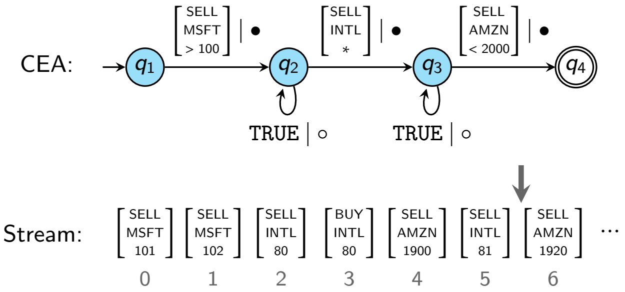




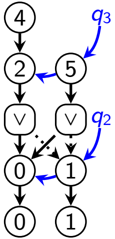
Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

Evaluation Algorithm by Example



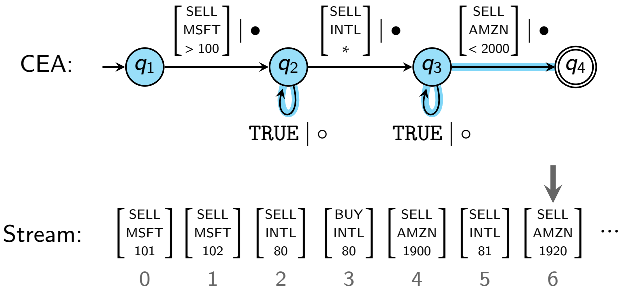
tECS:



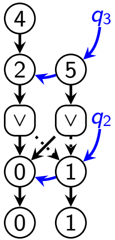
Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\mathrm{CEA}|)$, implying that we only take time $\mathcal{O}(|\mathrm{CEA}|)$ per event. This is constant in data complexity.

Evaluation Algorithm by Example



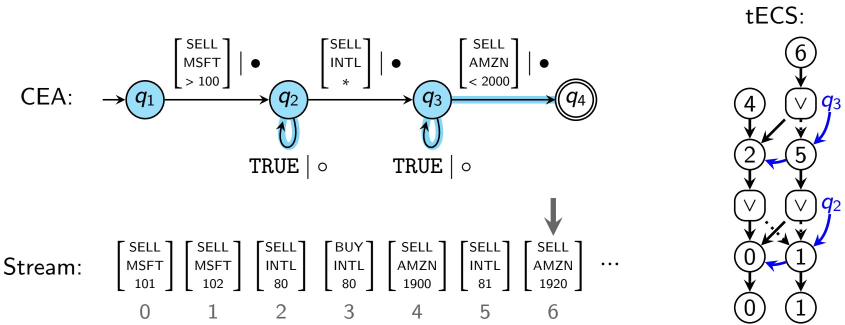
tECS:



Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\mathrm{CEA}|)$, implying that we only take time $\mathcal{O}(|\mathrm{CEA}|)$ per event. This is constant in data complexity.

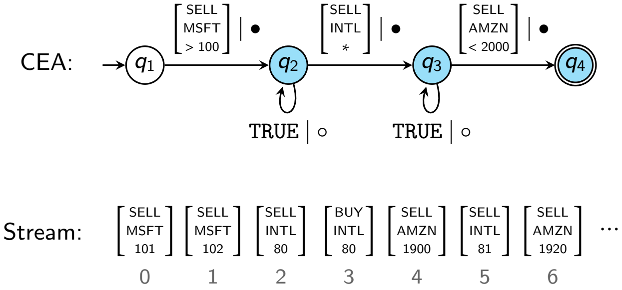
Evaluation Algorithm by Example

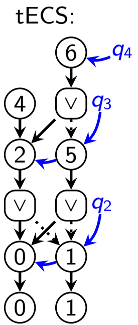


Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

Evaluation Algorithm by Example

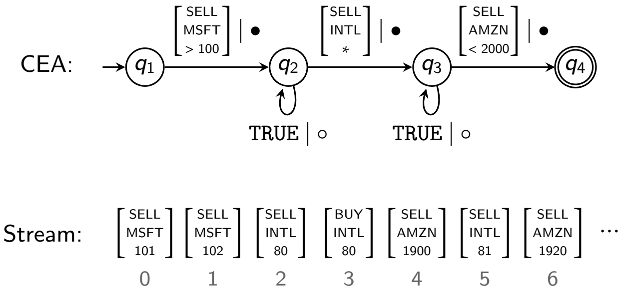


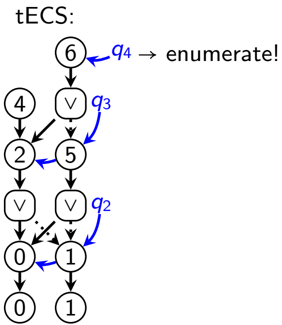


Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

Evaluation Algorithm by Example





Algorithm crux:

- Incrementally build the (well-structured) tECS to represents all open complex events up to the current event.
- Maintain the set of *active* CEA states, and the open complex events they correspond to.
- Crucially, all bookkeeping is $\mathcal{O}(|\text{CEA}|)$, implying that we only take time $\mathcal{O}(|\text{CEA}|)$ per event. This is constant in data complexity.

Outline

A logic for CER

An automaton model for CER

Evaluation algorithm

The CORE complex event recognition engine

Open questions

CORE: **CO**mplex event **R**ecognition **E**ngine

An **open-source implementation** [1] of our approach.

[1] `https://github.com/CORE-cer/CORE`

CORE: **CO**mplex event **R**ecognition **E**ngine

An **open-source implementation** [1] of our approach.

1. Practical query language (CEQL) based on unary CEL.
2. Evaluation in constant update-time and output-linear delay, based on CEA.
3. CORE's performance is stable w.r.t query and time-window size.
4. CORE outperforms existing systems by up to 5 orders of magnitude.

[1] `https://github.com/CORE-cer/CORE`

CEQL: Complex Event Query language

CEQL: Complex Event Query language

SELECT	< list-of-variables >
FROM	< list-of-streams >
WHERE	< CEL-formula >
FILTER	< list-of-filters >
[PARTITION BY	< list-of-attributes >]
[WITHIN	< time-value >]

CEQL: Complex Event Query language

SELECT	< list-of-variables >
FROM	< list-of-streams >
WHERE	< CEL-formula >
FILTER	< list-of-filters >
[PARTITION BY	< list-of-attributes >]
[WITHIN	< time-value >]

Examples (Stock Market)

1.
```
SELECT   * FROM Stocks
WHERE    SELL as msft; SELL as intel; SELL as amzn
FILTER   msft[name="MSFT"] AND msft[price > 100]
         AND intel[name="INTL"]
         AND amzn[name="AMZN"] AND amzn[price < 2000]
```

CEQL: Complex Event Query language

SELECT	< list-of-variables >
FROM	< list-of-streams >
WHERE	< CEL-formula >
FILTER	< list-of-filters >
[PARTITION BY	< list-of-attributes >]
[WITHIN	< time-value >]

Examples (Stock Market)

1.
```
SELECT   * FROM Stocks
WHERE    SELL as msft; SELL as intel; SELL as amzn
FILTER   msft[name="MSFT"] AND msft[price > 100]
         AND intel[name="INTL"]
         AND amzn[name="AMZN"] AND amzn[price < 2000]
```

Stream: $\begin{bmatrix} \text{SELL} \\ \text{MSFT} \\ 101 \end{bmatrix} \begin{bmatrix} \text{SELL} \\ \text{MSFT} \\ 102 \end{bmatrix} \begin{bmatrix} \text{SELL} \\ \text{INTL} \\ 80 \end{bmatrix} \begin{bmatrix} \text{BUY} \\ \text{INTL} \\ 80 \end{bmatrix} \begin{bmatrix} \text{SELL} \\ \text{AMZN} \\ 1900 \end{bmatrix} \begin{bmatrix} \text{BUY} \\ \text{INTL} \\ 81 \end{bmatrix} \begin{bmatrix} \text{BUY} \\ \text{AMZN} \\ 1920 \end{bmatrix} \cdots \quad \begin{matrix} \text{(type)} \\ \text{(name)} \\ \text{(price)} \end{matrix}$

CEQL: Complex Event Query language

SELECT	< list-of-variables >
FROM	< list-of-streams >
WHERE	< CEL-formula >
FILTER	< list-of-filters >
[PARTITION BY	< list-of-attributes >]
[WITHIN	< time-value >]

Examples (Stock Market)

1.
```
SELECT   * FROM Stocks
WHERE    SELL as msft; SELL as intel; SELL as amzn
FILTER   msft[name="MSFT"] AND msft[price > 100]
         AND intel[name="INTL"]
         AND amzn[name="AMZN"] AND amzn[price < 2000]
```

Stream:

SELL	SELL	**SELL**	BUY	**SELL**	BUY	BUY	...	(type)
MSFT	MSFT	**INTL**	INTL	**AMZN**	INTL	AMZN		(name)
101	102	**80**	80	**1900**	81	1920		(price)

CEQL: Complex Event Query language

SELECT	< list-of-variables >
FROM	< list-of-streams >
WHERE	< CEL-formula >
FILTER	< list-of-filters >
[PARTITION BY	< list-of-attributes >]
[WITHIN	< time-value >]

Examples (Stock Market)

1.
```
SELECT    * FROM Stocks
WHERE     SELL as msft; SELL as intel; SELL as amzn
FILTER    msft[name="MSFT"] AND msft[price > 100]
          AND intel[name="INTL"]
          AND amzn[name="AMZN"] AND amzn[price < 2000]
```

Stream:

SELL	**SELL**	**SELL**	BUY	**SELL**	BUY	BUY	...	(type)
MSFT	**MSFT**	**INTL**	INTL	**AMZN**	INTL	AMZN		(name)
101	**102**	**80**	80	**1900**	81	1920		(price)

CEQL: Complex Event Query language

SELECT	< list-of-variables >
FROM	< list-of-streams >
WHERE	< CEL-formula >
FILTER	< list-of-filters >
[**PARTITION BY**	< list-of-attributes >]
[**WITHIN**	< time-value >]

CEQL: Complex Event Query language

SELECT	< list-of-variables >
FROM	< list-of-streams >
WHERE	< CEL-formula >
FILTER	< list-of-filters >
[**PARTITION BY**	< list-of-attributes >]
[**WITHIN**	< time-value >]

Examples (Stock Market)

2.
```
SELECT        s, b  FROM Stocks
WHERE         (BUY or SELL) as s; (BUY or SELL) as b
PARTITION BY  [name]
WITHIN        5 minute
```

CEQL: Complex Event Query language

SELECT	< list-of-variables >
FROM	< list-of-streams >
WHERE	< CEL-formula >
FILTER	< list-of-filters >
[**PARTITION BY**	< list-of-attributes >]
[**WITHIN**	< time-value >]

Examples (Stock Market)

2.
```
SELECT         s, b  FROM Stocks
WHERE          (BUY or SELL) as s; (BUY or SELL) as b
PARTITION BY   [name]
WITHIN         5 minute
```

Stream:

SELL	SELL	SELL	BUY	SELL	BUY	BUY	...	(type)
MSFT	MSFT	INTL	INTL	AMZN	INTL	AMZN		(name)
101	102	80	80	1900	81	1920		(price)
10:00	10:02	10:10	10:14	10:25	10:30	10:33		(time)

CEQL: Complex Event Query language

SELECT	< list-of-variables >
FROM	< list-of-streams >
WHERE	< CEL-formula >
FILTER	< list-of-filters >
[**PARTITION BY**	< list-of-attributes >]
[**WITHIN**	< time-value >]

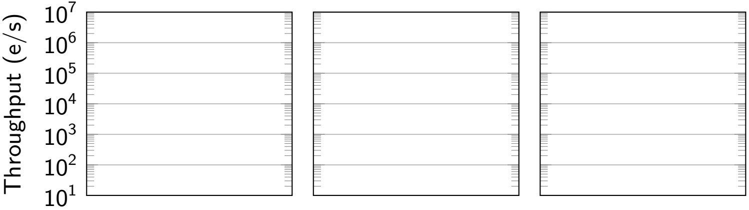
Examples (Stock Market)

2.
```
SELECT         s, b  FROM Stocks
WHERE          (BUY or SELL) as s; (BUY or SELL) as b
PARTITION BY   [name]
WITHIN         5 minute
```

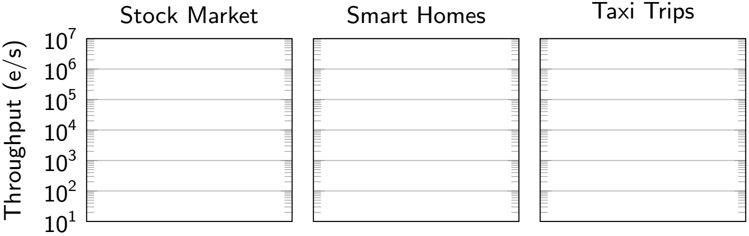
Stream:

SELL	**SELL**	SELL	BUY	SELL	BUY	BUY	...	(type)
MSFT	**MSFT**	INTL	INTL	AMZN	INTL	AMZN		(name)
101	**102**	80	80	1900	81	1920		(price)
10:00	**10:02**	10:10	10:14	10:25	10:30	10:33		(time)

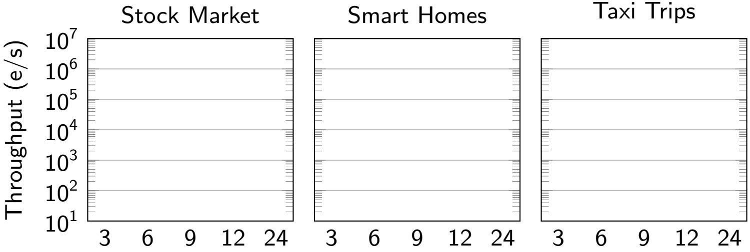
Experiments: Sequence queries



Experiments: Sequence queries



Experiments: Sequence queries



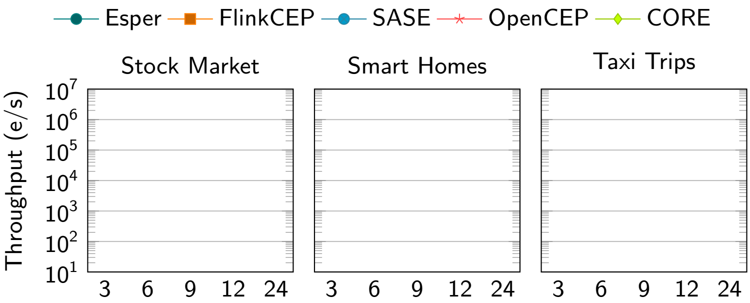

```
SELECT   * FROM Dataset
WHERE    A1 ; A2 ; ...  ; An
FILTER   A1[filter_1] AND ...  AND An[filter_n]
WITHIN   T
```

Experiments: Sequence queries


```
SELECT   * FROM Dataset
WHERE    A1 ; A2 ; ...  ; An
FILTER   A1[filter_1] AND ...  AND An[filter_n]
WITHIN   T
```

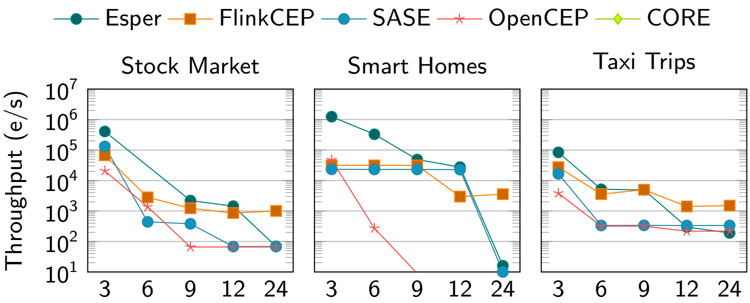
We use sequences of length $n = 3, 6, 9, 12, 24$.

Experiments: Sequence queries

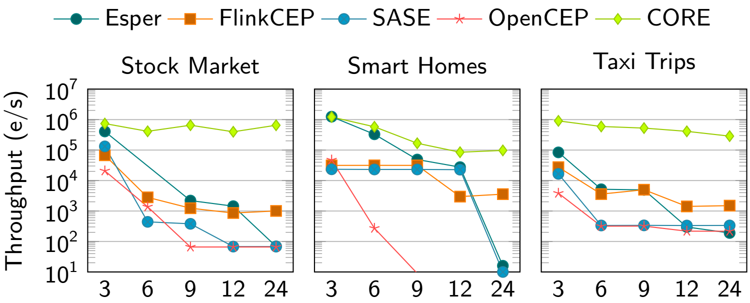


1. Esper (*industry*)
2. FlinkCEP (*industry*)
3. SASE (*academy*)
4. OpenCEP (*academy*)
5. CORE

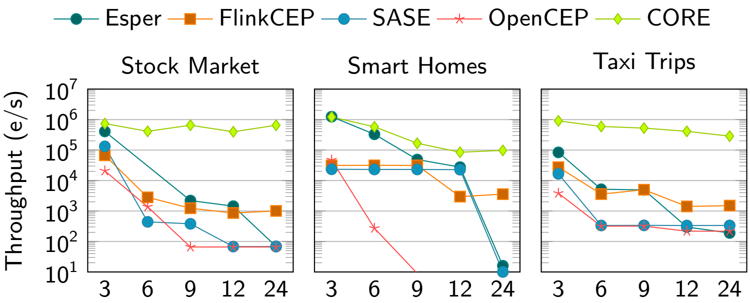
Experiments: Sequence queries



Experiments: Sequence queries

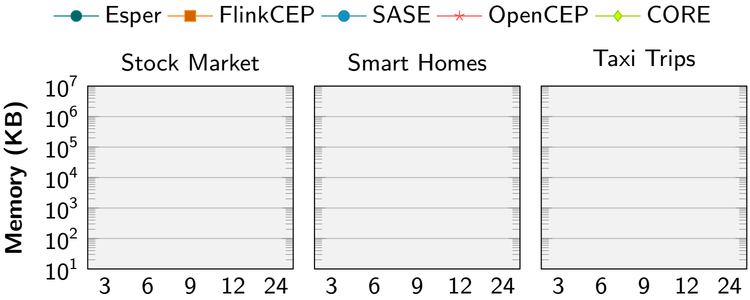


Experiments: Sequence queries

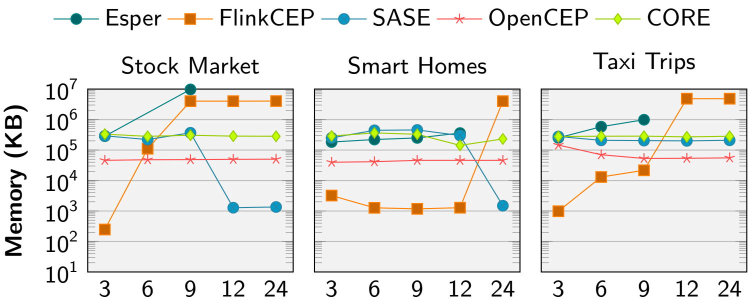


CORE is up to **4 orders of magnitude faster** than other systems

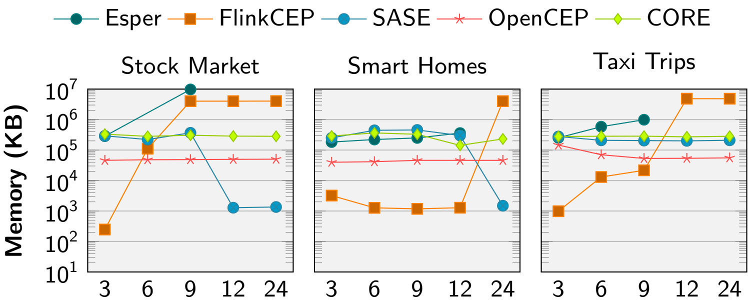
Experiments: Sequence queries (memory)



Experiments: Sequence queries (memory)

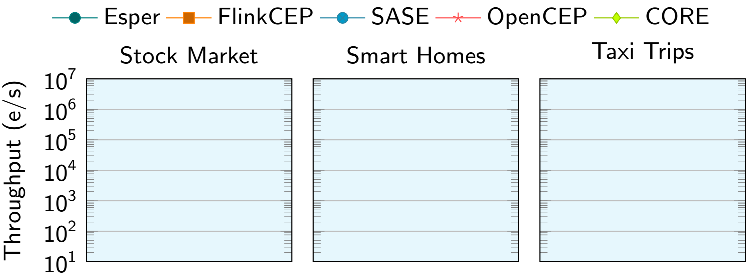


Experiments: Sequence queries (memory)

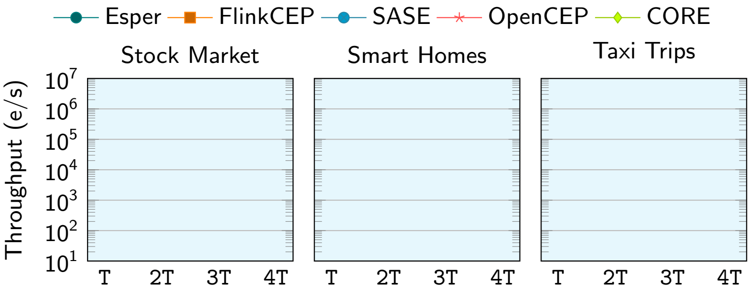


CORE is **stable** in the memory usage

Experiments: Window queries

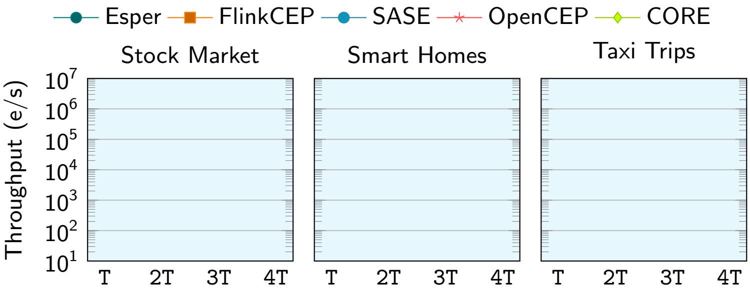


Experiments: Window queries




```
SELECT   * FROM Dataset
WHERE    A1 ; A2 ; A3
FILTER   A1[filter₁] AND A2[filter₂] AND A3[filter₃]
WITHIN   X
```

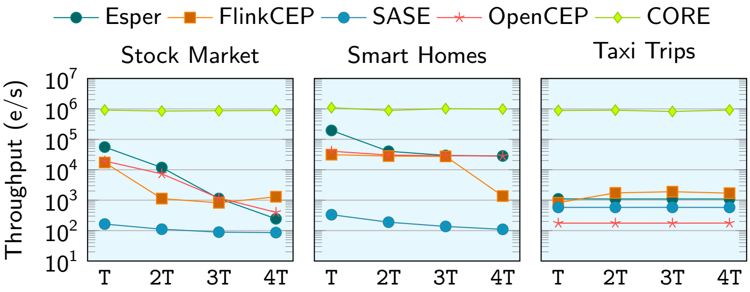
Experiments: Window queries



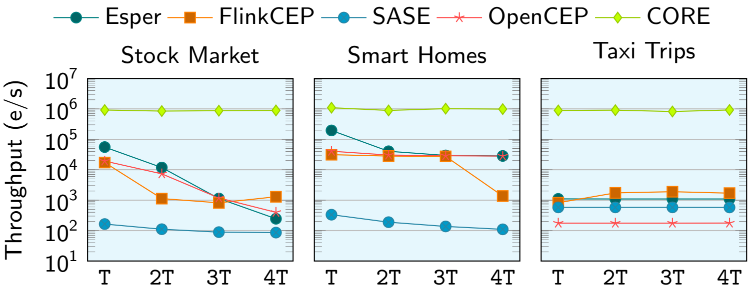

```
SELECT   * FROM Dataset
WHERE    A1 ; A2 ; A3
FILTER   A1[filter₁] AND A2[filter₂] AND A3[filter₃]
WITHIN   X
```

We use time-windows **size** `X` = `T`, `2T`, `3T`, `4T`.

Experiments: Window queries



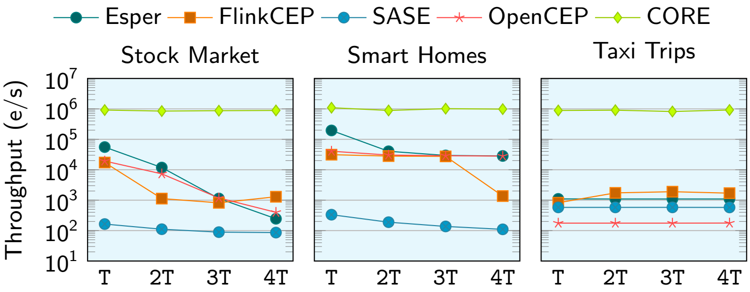
Experiments: Window queries



Conclusions

1. CORE is orders of magnitude faster than other systems.

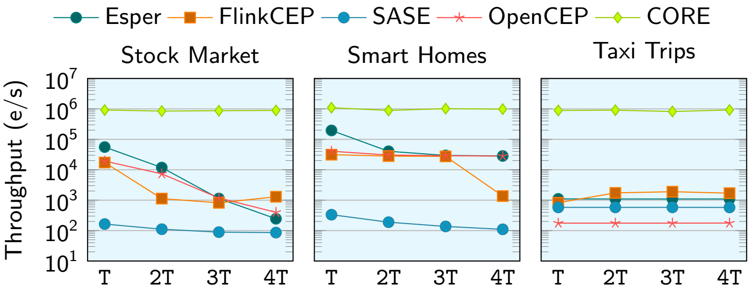
Experiments: Window queries



Conclusions

1. CORE is orders of magnitude faster than other systems.
2. CORE is **not affected** by the **query** or **time-windows** size.

Experiments: Window queries



In the paper [1], we show similar results with other query workloads

[1] M. Bucchi, A. Grez, A. Quintana, C. Riveros, and S. Vansummeren
"CORE: a Complex Event Recognition Engine", VLDB 2022.

Outline

A logic for CER

An automaton model for CER

Evaluation algorithm

The CORE complex event recognition engine

Open questions

Time Model

Time Model

Limitation: No out-of-order events

- Time is implicit, given by arrival order
- Crucial property for CEA evaluation:
  Events arrive in timestamp order

Time Model

Limitation: No out-of-order events

- Time is implicit, given by arrival order
- Crucial property for CEA evaluation:
  Events arrive in timestamp order

Open question: What is the impact of out-of-order events on

- Language design and expressiveness ?
- Evaluation model (CEA) and complexity ?

Event correlation

Limitation: CORE and CEQL are based on **unary** CEL

Event correlation

Limitation: CORE and CEQL are based on **unary** CEL

- Unary CEL does not allow event **correlation**.

Example: unsupported

$$\varphi = (B \, ; \, S) \; \texttt{FILTER} \; B[\mathrm{id}] = S[\mathrm{id}] \wedge B[\mathit{volume}] > S[\mathit{volume}]$$

Event correlation

Limitation: CORE and CEQL are based on **unary** CEL

- Unary CEL does not allow event **correlation**.
- ... partially solved by PARTITION BY in CEQL for equality in limited cases.

Example: unsupported

$$\varphi = (B\,;\,S)\ \texttt{FILTER}\ B[\mathrm{id}] = S[\mathrm{id}] \wedge B[\mathit{volume}] > S[\mathit{volume}]$$

Event correlation

Limitation: CORE and CEQL are based on **unary** CEL

- Unary CEL does not allow event **correlation**.
- ... partially solved by PARTITION BY in CEQL for equality in limited cases.

Example: unsupported

$$\varphi = (B \,;\, S) \texttt{ FILTER } B[\mathrm{id}] = S[\mathrm{id}] \wedge B[\mathit{volume}] > S[\mathit{volume}]$$

Open questions:

- What is the impact of moving to k-ary predicates, $k > 1$ on Language expressiveness ?
- What is the right computational model (à la CEA) with binary predicates ?
- How does this affect complexity?

Processing versus recognition

Limitation: CORE, CEQL, and CEL focus on complex event **recognition**

Processing versus recognition

Limitation: CORE, CEQL, and CEL focus on complex event **recognition**

Other features in the literature that focus on *processing* of complex events are not supported:

- aggregation
- integration of non-event data sources
- parallel or distributed execution

Processing versus recognition

Limitation: CORE, CEQL, and CEL focus on complex event **recognition**

Other features in the literature that focus on *processing* of complex events are not supported:

- aggregation
- integration of non-event data sources
- parallel or distributed execution

Open questions:

- What is the right language for CER + aggregation?
- What is the right computational model (à la CEA) in the presence of aggregation?
- How does aggregation affect evaluation complexity?

Getting to the CORE of Complex Event Recognition

Stijn Vansummeren
UHasselt, Data Science Institute