Early Detection of Business Rule Violations

Isaac Mackey

University of California, Santa Barbara

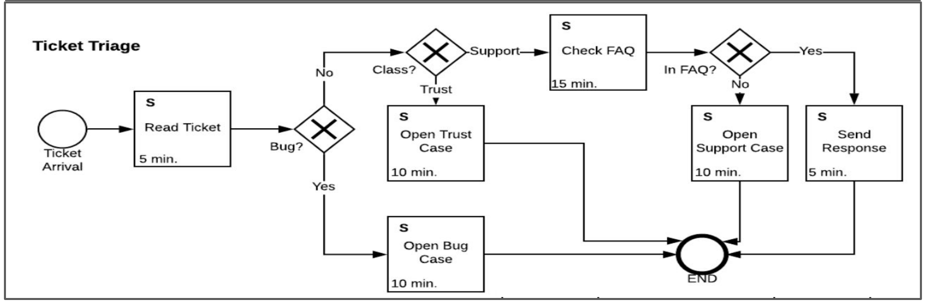
PhD Committee

Jianwen Su (Chair), Tevfik Bultan, Daniel Lokshtanov

Events Streams Must Satisfy Business Rules

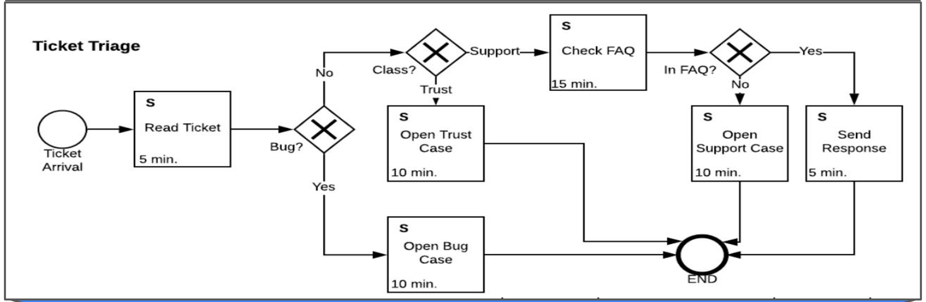
Events Streams Must Satisfy Business Rules

Business Workflows



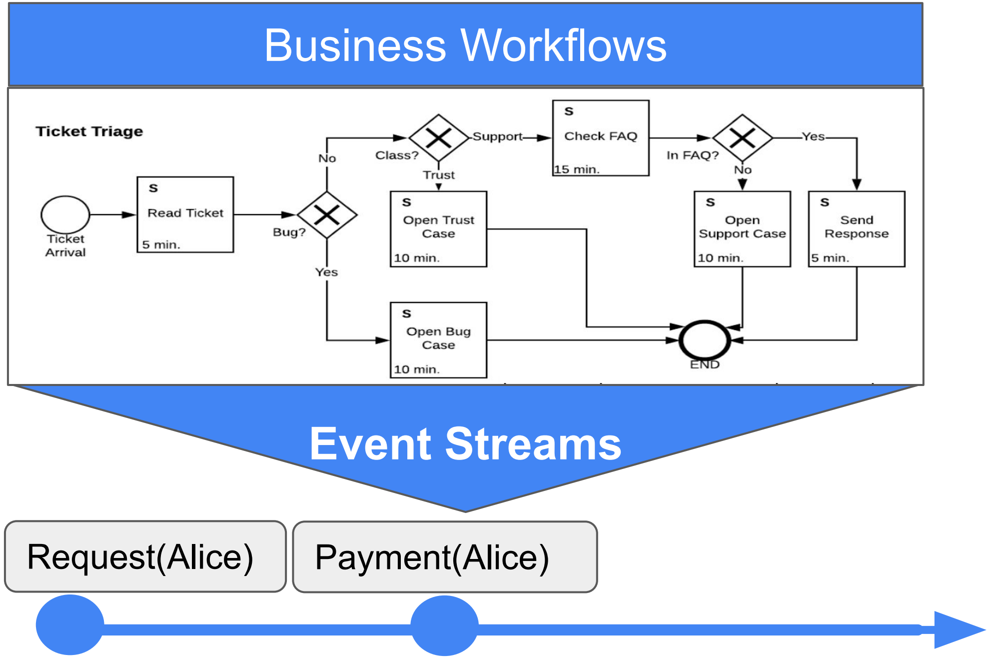
Events Streams Must Satisfy Business Rules

Business Workflows

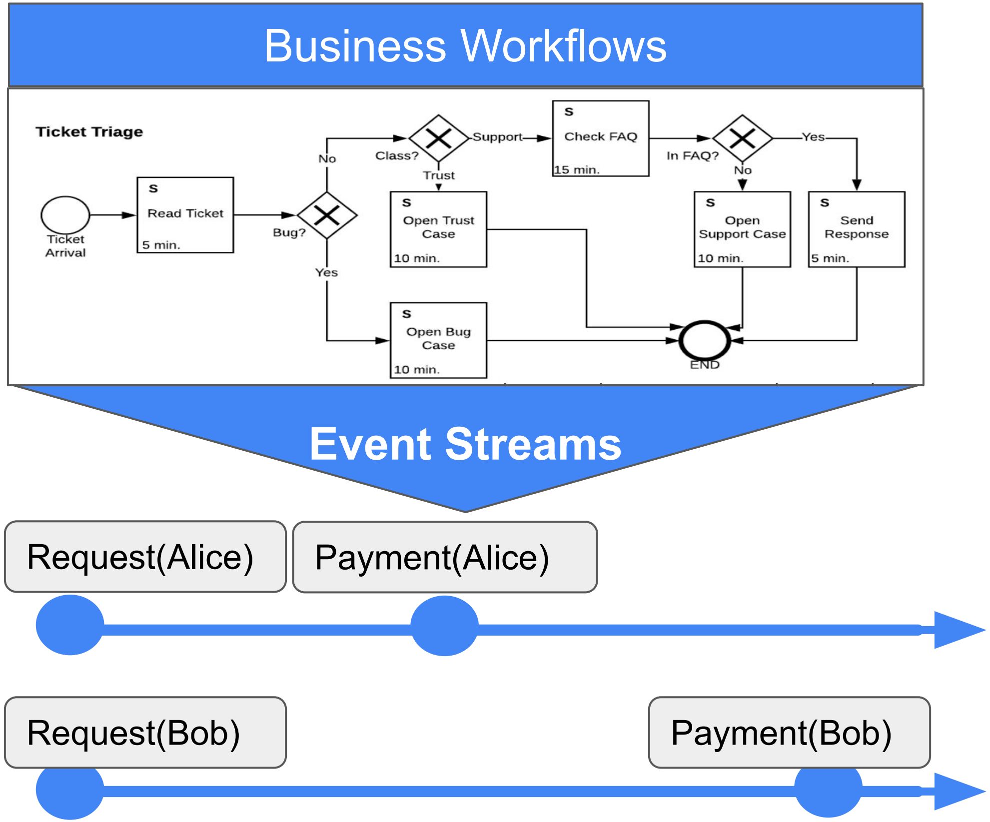


Event Streams

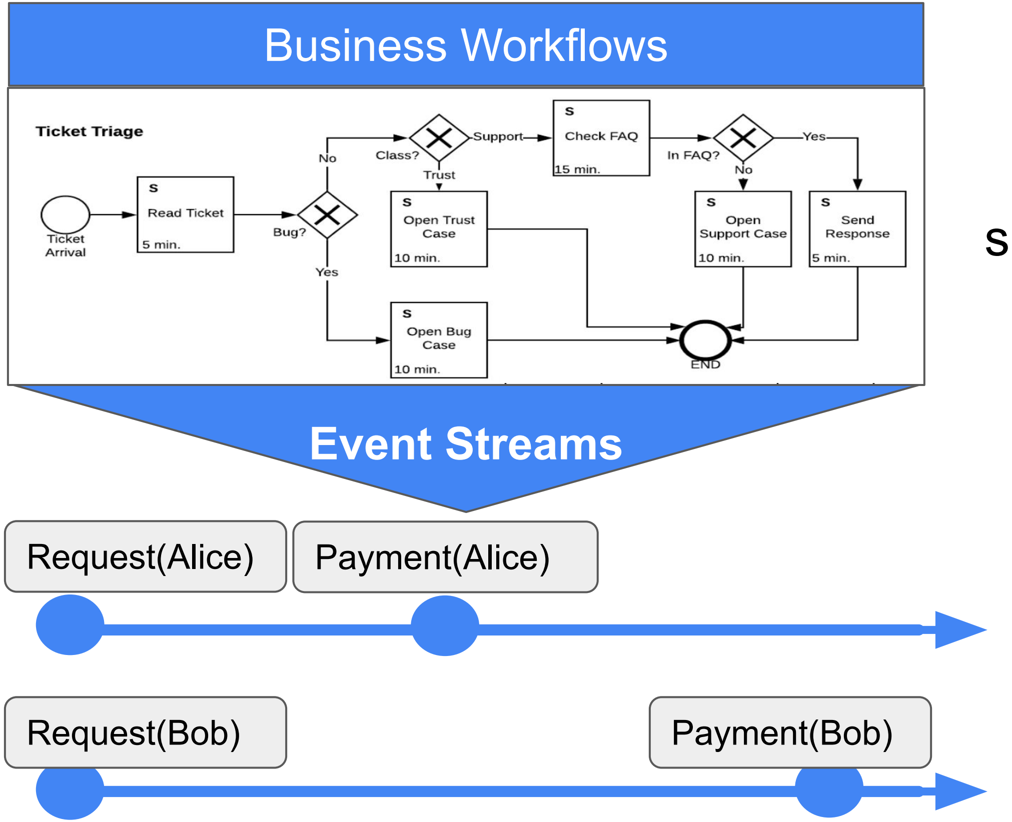
Events Streams Must Satisfy Business Rules



Events Streams Must Satisfy Business Rules

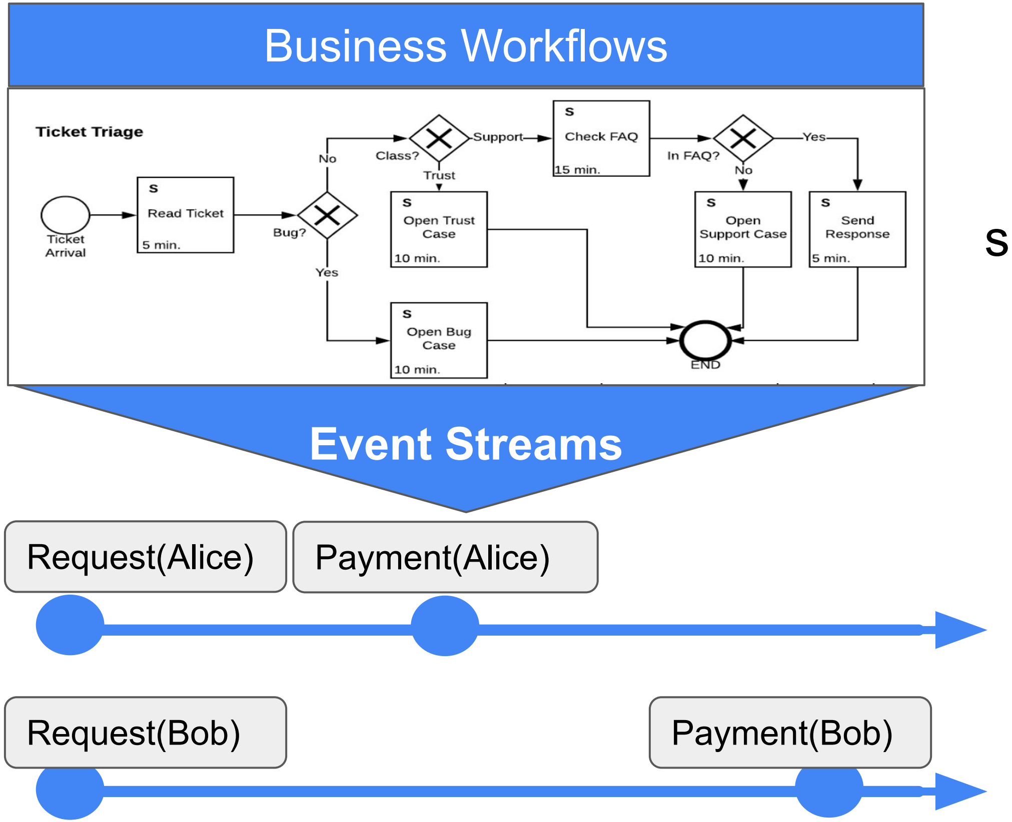


Events Streams Must Satisfy Business Rules



subject to **Business Rules**

Events Streams Must Satisfy Business Rules

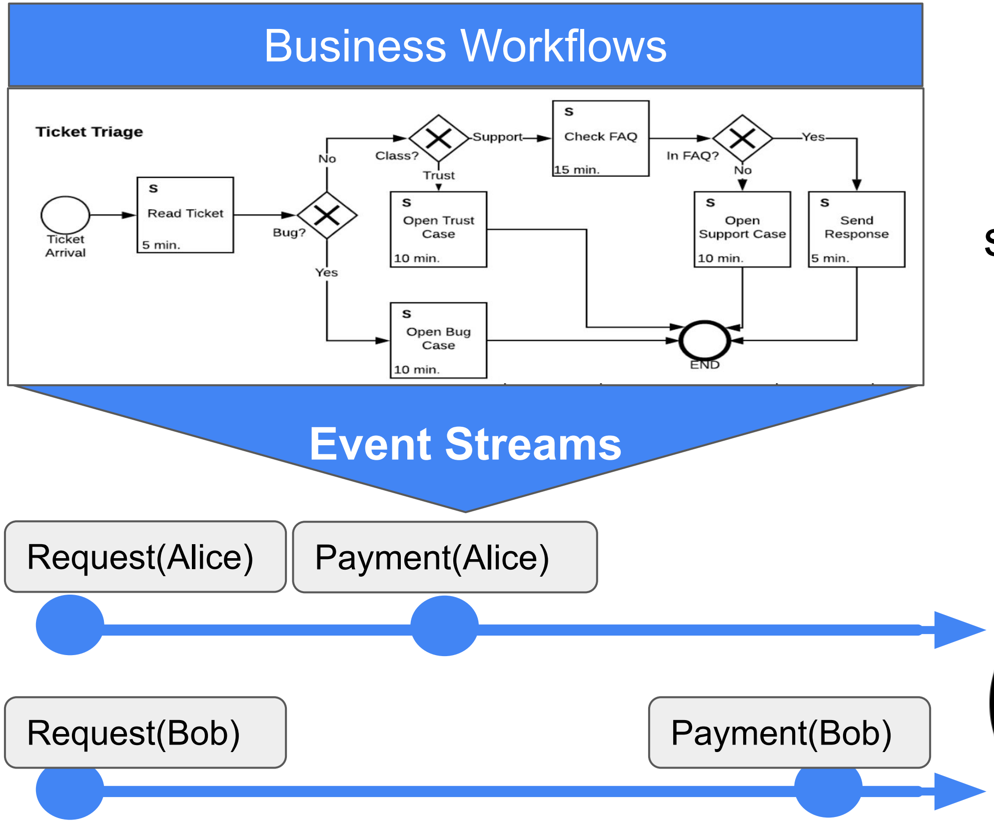


subject to

Business Rules

- Organizational goals
- Resource limits
- Safety regulations
- ...

Events Streams Must Satisfy Business Rules



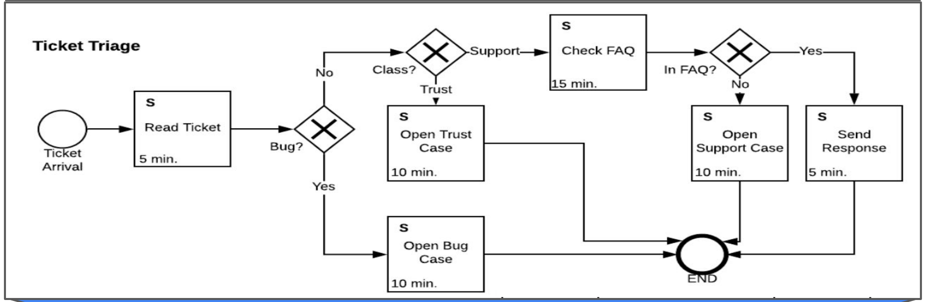
subject to

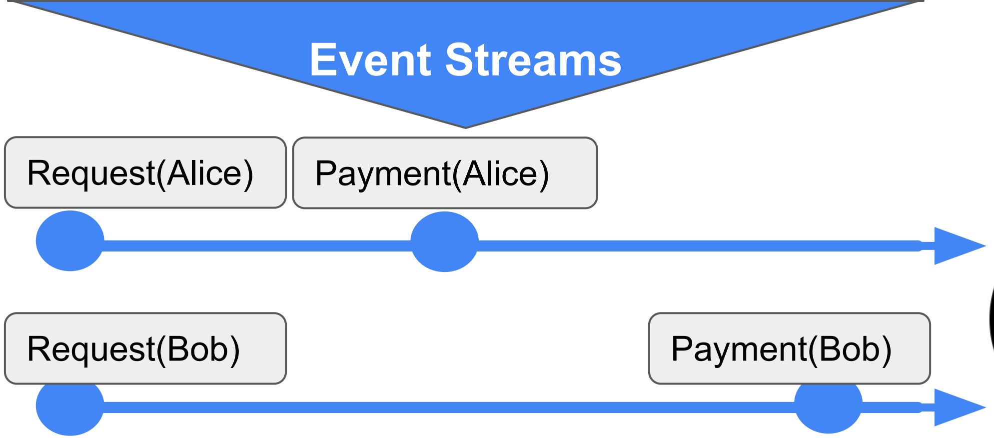
Business Rules

- Organizational goals
- Resource limits
- Safety regulations
- ...

Events Streams Must Satisfy Business Rules

Business Workflows





subject to

Business Rules

- Organizational goals
- Resource limits
- Safety regulations
- ...

SATISFIES RULES

VIOLATION, e.g., Late Payment

Rule Specification Language

Rule Specification Language

Request(user u)@x, Approval(user u)@y, x+1 ≤ y

→

Payment(user v)@z, Receipt(user v)@w, z ≤ y+10, z ≤ w ≤ z+2

Rule Specification Language

$$\overbrace{\text{Request(user u)@x}}^{\textit{event}}\text{, Approval(user u)@y, x+1} \le \text{y}$$

→

Payment(user v)@z, Receipt(user v)@w, z ≤ y+10, z ≤ w ≤ z+2

Rule Specification Language

$\overbrace{\text{Request(user u)@x}}^{\textit{event}}$, Approval(user u)@y, $\overbrace{x+1 \le y}^{\textit{quantitative time gap}}$

→

Payment(user v)@z, Receipt(user v)@w, $z \le y+10$, $z \le w \le z+2$

Rule Specification Language

$$\overbrace{\text{Request(user u)@x}}^{\textit{event}}, \text{ Approval(user u)@y}, \ \overbrace{x+1 \le y}^{\textit{quantitative time gap}}$$

$$\rightarrow$$

$$\text{Payment(user v)@z, Receipt(user v)@w}, \ z \le y+10, \ z \le w \le z+2$$

data variable *time variable*

Rule Specification Language

$\overbrace{\text{Request(user u)@x}}^{\textit{event}}$, Approval(user u)@y, $\overbrace{x+1 \le y}^{\textit{quantitative time gap}}$ *body*

$\rightarrow$

Payment(user v)@z, Receipt(user v)@w, $z \le y+10$, $z \le w \le z+2$ *head*

data variable *time variable*

Rule Specification Language

$\overbrace{\text{Request(user u)@x}}^{\textit{event}}$, Approval(user u)@y, $\overbrace{x+1 \leq y}^{\textit{quantitative time gap}}$ *body*

$\rightarrow$

Payment(user v)@z, Receipt(user v)@w, z ≤ y+10, z ≤ w ≤ z+2 *head*

data variable *time variable*

Request(Alice)@1

Rule Specification Language

$$\overbrace{\text{Request(user u)@x}}^{\textit{event}}\text{, Approval(user u)@y, }\overbrace{x+1 \le y}^{\textit{quantitative time gap}} \qquad \textit{body}$$

$$\rightarrow$$

$$\text{Payment(user v)@z, Receipt(user v)@w, } z \le y+10,\ z \le w \le z+2 \qquad \textit{head}$$

data variable *time variable*

Request(Alice)@1

Request(Alice)@1 — Approval(Alice)@2 — Payment(Bob)@11 — Receipt(Bob)@12

Detecting Rule Violations at the Earliest Possible Time

Detecting Rule Violations at the Earliest Possible Time

Request(user u)@x, Approval(user u)@y, $x+1 \leq y$

$\rightarrow$ Payment(user v)@z, Receipt(user v)@w, $z \leq y+10$, $z \leq w \leq z+2$

Detecting Rule Violations at the Earliest Possible Time

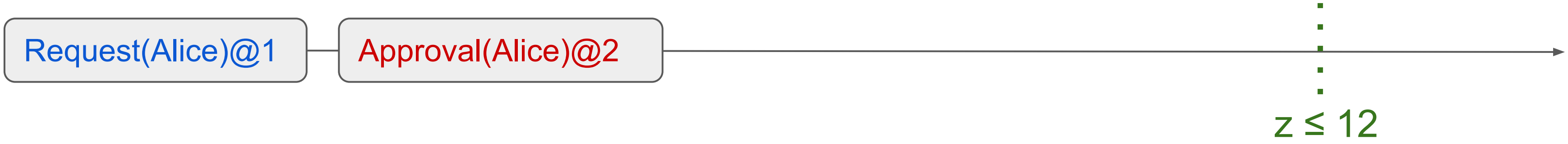
Request(user u)@x, Approval(user u)@y, $x+1 \leq y$

$\rightarrow$ Payment(user v)@z, Receipt(user v)@w, $z \leq y+10$, $z \leq w \leq z+2$

Detecting Rule Violations at the Earliest Possible Time

Request(user u)@x, Approval(user u)@y, $x+1 \leq y$

→ Payment(user v)@z, Receipt(user v)@w, $z \leq y+10$, $z \leq w \leq z+2$



Detecting Rule Violations at the Earliest Possible Time

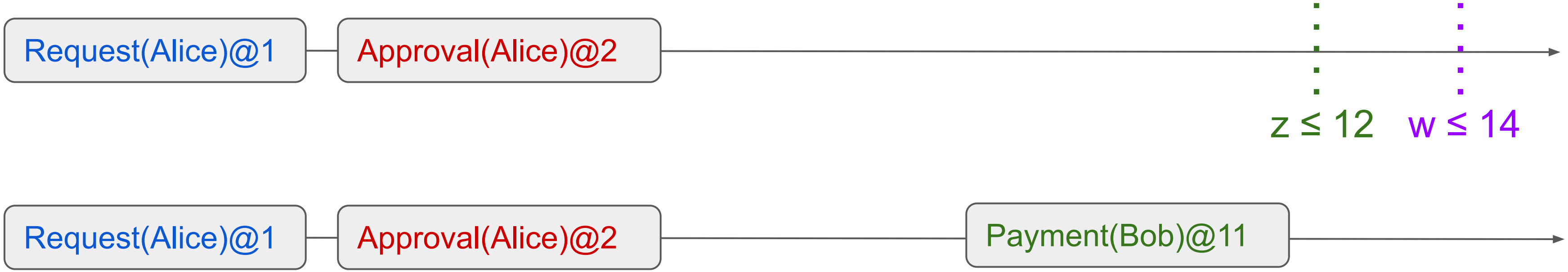
Request(user u)@x, Approval(user u)@y, x+1 ≤ y

→ Payment(user v)@z, Receipt(user v)@w, z ≤ y+10, z ≤ w ≤ z+2

Detecting Rule Violations at the Earliest Possible Time

Request(user u)@x, Approval(user u)@y, $x+1 \leq y$

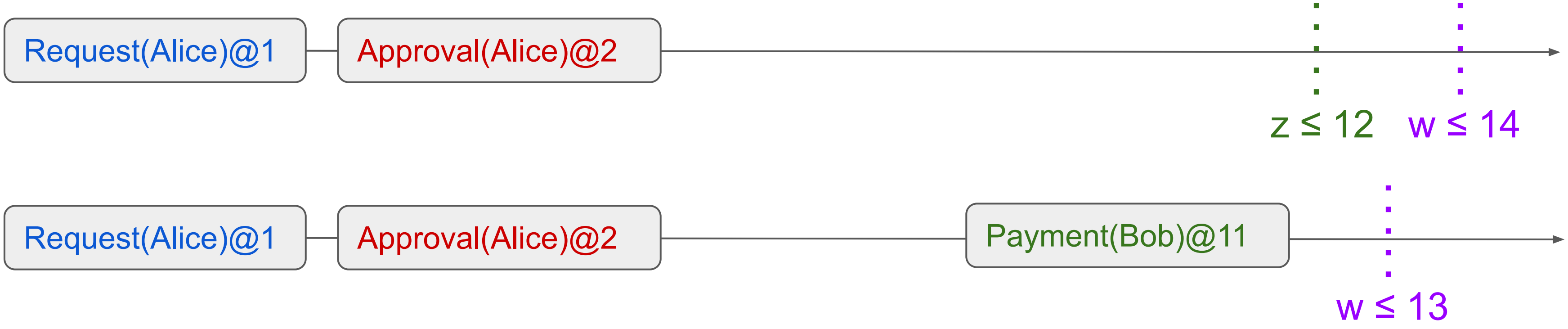
$\rightarrow$ Payment(user v)@z, Receipt(user v)@w, $z \leq y+10$, $z \leq w \leq z+2$



Detecting Rule Violations at the Earliest Possible Time

$\text{Request(user u)@}x$, $\text{Approval(user u)@}y$, $x+1 \leq y$

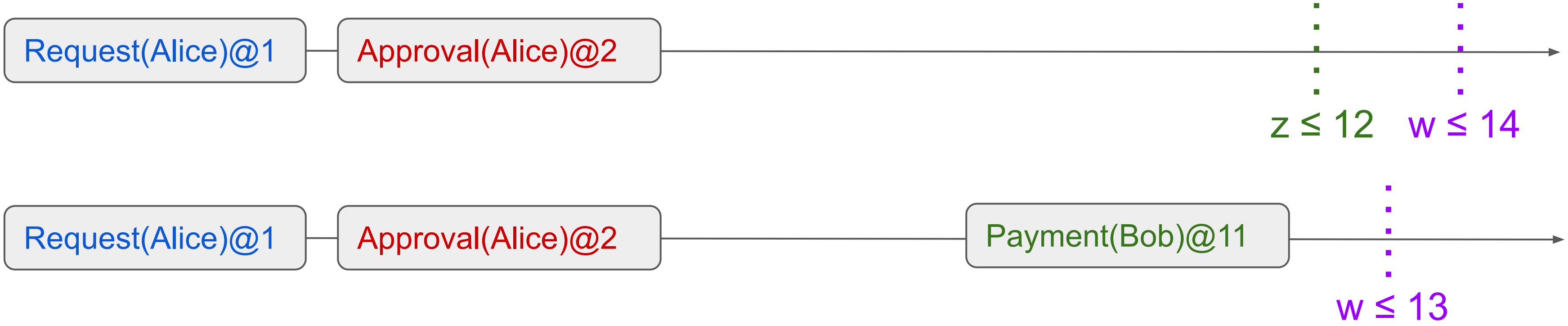
$\rightarrow$ $\text{Payment(user v)@}z$, $\text{Receipt(user v)@}w$, $z \leq y+10$, $z \leq w \leq z+2$



Detecting Rule Violations at the Earliest Possible Time

Request(user u)@x, Approval(user u)@y, $x+1 \le y$

$\rightarrow$ Payment(user v)@z, Receipt(user v)@w, $z \le y+10$, $z \le w \le z+2$

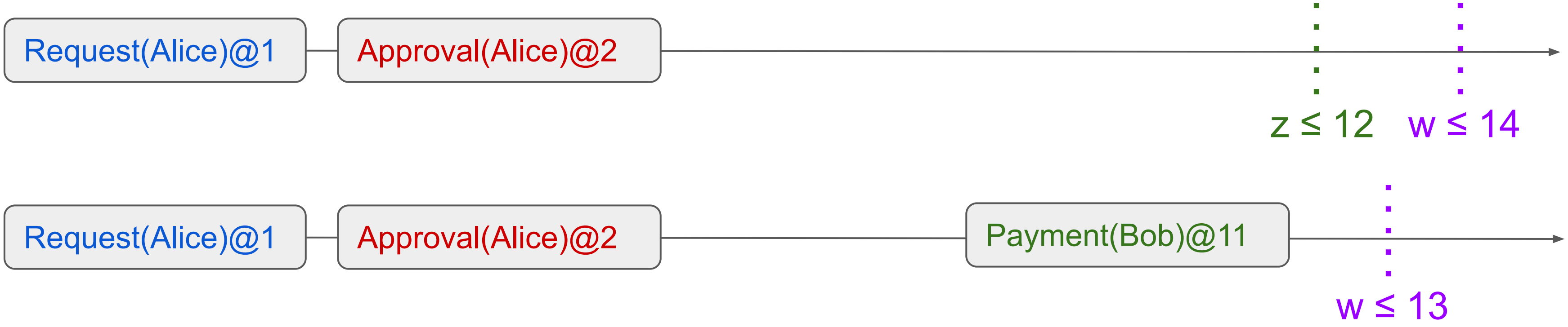


An event stream *e* is a ***violation*** of a rule (set of rules) *R* if no extension of *e* satisfies (each rule in) *R*.

Detecting Rule Violations at the Earliest Possible Time

Request(user u)@x, Approval(user u)@y, $x+1 \leq y$

$\rightarrow$ Payment(user v)@z, Receipt(user v)@w, $z \leq y+10$, $z \leq w \leq z+2$



An event stream *e* is a ***violation*** of a rule (set of rules) *R* if no extension of *e* satisfies (each rule in) *R*.

Can we detect violations at the earliest possible time?

Outline

Rules

1. without Data (Mackey & Su, Info. Sys., 2023), Comparison with Other Work

with Data

2. Individual Rules (Mackey et al., TIME 2022)

Sets of Rules

Acyclic Sets

Chase Process

(Mackey & Su, Info. & Comp., in revision, 2023)

3. with Aggregation Functions on Time Windows

4. Unrestricted Sets (Early Detection is Impossible)

Outline

Rules

1. **without Data (Mackey & Su, Info. Sys., 2023), Comparison with Other Work**

with Data

2. Individual Rules (Mackey et al., TIME 2022)

Sets of Rules

Acyclic Sets

Chase Process

(Mackey & Su, Info. & Comp., in revision, 2023)

3. with Aggregation Functions on Time Windows

4. Unrestricted Sets (Early Detection is Impossible)

Kamp's Theorem Offers Translation for Dataless Rules

Kamp's Theorem Offers Translation for Dataless Rules

Theorem. (Kamp, thesis, 1968): Given any dataless rule, there is an equivalent LTL formula.

Kamp's Theorem Offers Translation for Dataless Rules

Theorem. (Kamp, thesis, 1968): Given any dataless rule, there is an equivalent LTL formula.

Latest proof (Rabinovich, LMCS, 2014) provides translation whose size is *hyper-exponential* in size of input.

Kamp's Theorem Offers Translation for Dataless Rules

Theorem. (Kamp, thesis, 1968): Given any dataless rule, there is an equivalent LTL formula.

Latest proof (Rabinovich, LMCS, 2014) provides translation whose size is *hyper-exponential* in size of input.

(Mackey & Su, Info. Sys., 2023) provides translations...

Kamp's Theorem Offers Translation for Dataless Rules

Theorem. (Kamp, thesis, 1968): Given any dataless rule, there is an equivalent LTL formula.

Latest proof (Rabinovich, LMCS, 2014) provides translation whose size is *hyper-exponential* in size of input.

(Mackey & Su, Info. Sys., 2023) provides translations...

- For each singly-linked, acyclic, dataless rule r, an equivalent LTL formula whose size is *single-exponential* in |r|.

Kamp's Theorem Offers Translation for Dataless Rules

Theorem. (Kamp, thesis, 1968): Given any dataless rule, there is an equivalent LTL formula.

Latest proof (Rabinovich, LMCS, 2014) provides translation whose size is *hyper-exponential* in size of input.

(Mackey & Su, Info. Sys., 2023) provides translations...

- For each singly-linked, acyclic, dataless rule r, an equivalent LTL formula whose size is *single-exponential* in |r|.
- For each singly-linked, dataless rule r, an equivalent LTL formula whose size is *double-exponential* in |r|.

Outline

Rules

1. without Data (Mackey & Su, Info. Sys., 2023), Comparison with Other Work

with Data

2. Individual Rules (Mackey et al., TIME 2022)

Sets of Rules

Acyclic Sets

Chase Process

(Mackey & Su, Info. & Comp., in revision, 2023)

3. with Aggregation Functions on Time Windows

4. Unrestricted Sets (Early Detection is Impossible)

Interactions in Sets of Rules Creates Earlier Violations

Interactions in Sets of Rules Creates Earlier Violations

$$\begin{cases} \text{A:} \quad \mathrm{Request}(u)@x \rightarrow \mathrm{Payment}(u)@y,\ x \le y \\ \text{B:} \quad \mathrm{Request}(u)@x,\ \mathrm{Payment}(u)@y,\ x+5 \le y \rightarrow \mathrm{Approval}(v)@z,\ z < x \end{cases}$$

Interactions in Sets of Rules Creates Earlier Violations

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

Interactions in Sets of Rules Creates Earlier Violations

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

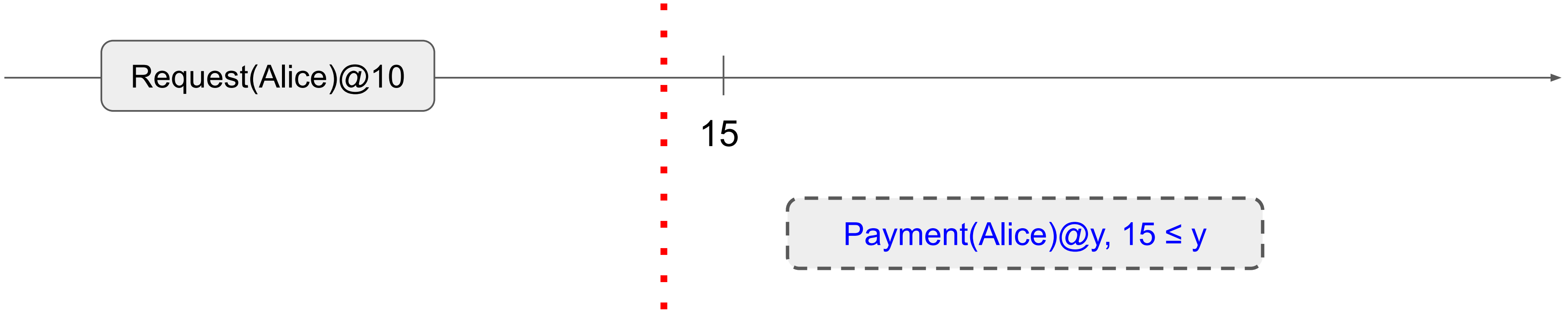
Request(Alice)@10

Payment(Alice)@y, 10 ≤ y

Interactions in Sets of Rules Creates Earlier Violations

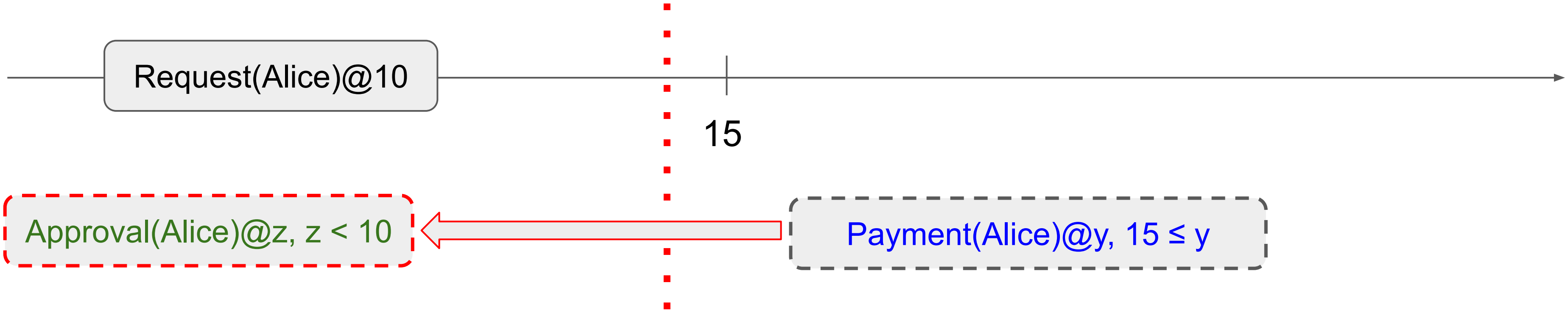
A: $\mathrm{Request}(u)@x \rightarrow \mathrm{Payment}(u)@y,\ x \le y$

B: $\mathrm{Request}(u)@x,\ \mathrm{Payment}(u)@y,\ x+5 \le y \rightarrow \mathrm{Approval}(v)@z,\ z < x$



Interactions in Sets of Rules Creates Earlier Violations

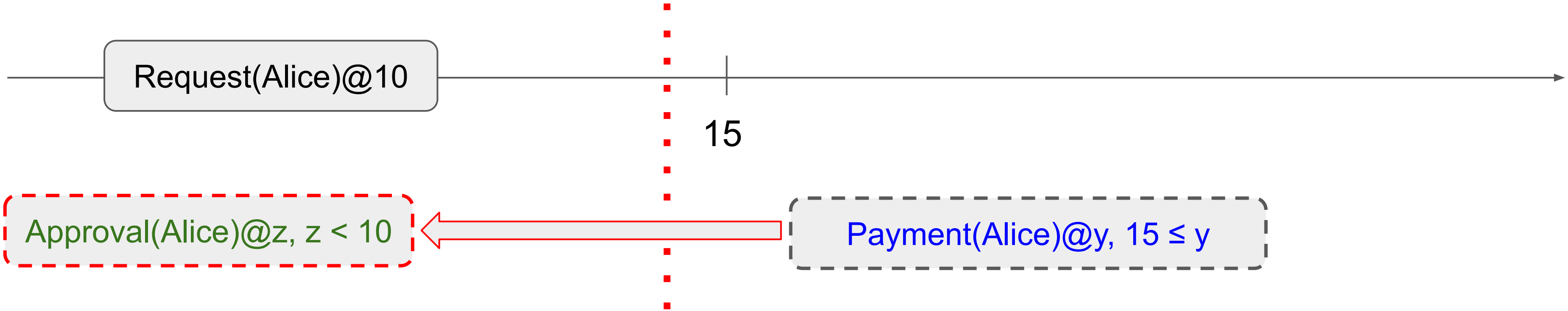
$$\begin{cases} \text{A:} \quad \text{Request}(u)@x \rightarrow \text{Payment}(u)@y,\ x \leq y \\ \text{B:} \quad \text{Request}(u)@x,\ \text{Payment}(u)@y,\ x+5 \leq y \rightarrow \text{Approval}(v)@z,\ z < x \end{cases}$$



Interactions in Sets of Rules Creates Earlier Violations

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

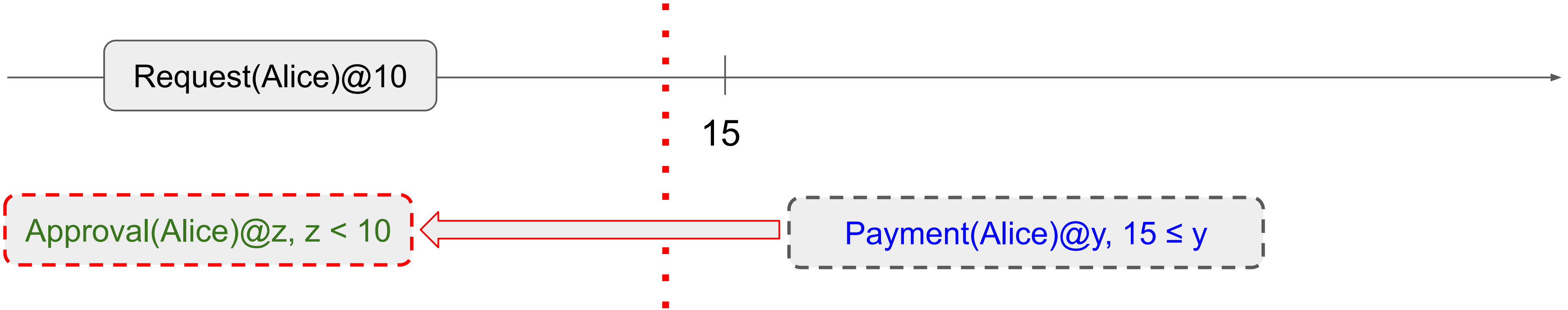


- No violation of A at *t* =15
- No violation of B at *t* =15

Interactions in Sets of Rules Creates Earlier Violations

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x



- No violation of A at *t* =15
- No violation of B at *t* =15

  ... but { A, B } is violated after *t* =15

Chase Creates Expected Events

Chase Creates Expected Events

$$\begin{cases} A: \quad \text{Request}(u)@x \rightarrow \text{Payment}(u)@y,\ x \leq y \\ B: \quad \text{Request}(u)@x,\ \text{Payment}(u)@y,\ x+5 \leq y \rightarrow \text{Approval}(v)@z,\ z < x \end{cases}$$

Chase Creates Expected Events

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

Request(Alice)@10

Chase Creates Expected Events

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

Request(Alice)@10

Chasing A with
(u=Alice, x=10)

Chase Creates Expected Events

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

Request(Alice)@10

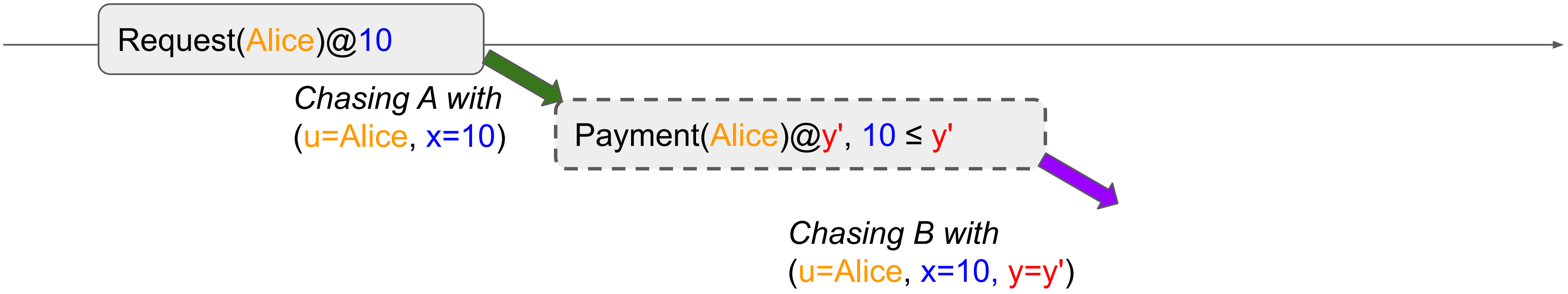
Chasing A with
(u=Alice, x=10)

Payment(Alice)@y', 10 ≤ y'

Chase Creates Expected Events

A: Request(u)@x → Payment(u)@y, x ≤ y

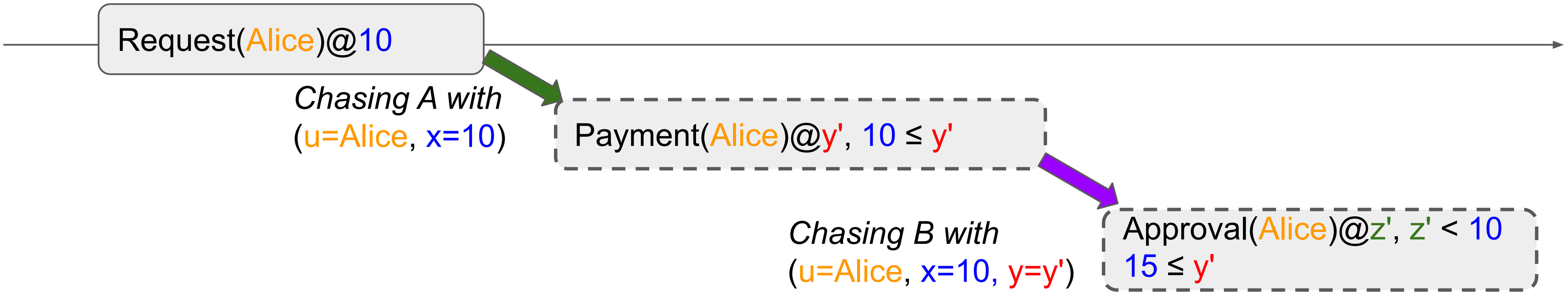
B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x



Chase Creates Expected Events

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x



Acyclic Set of Rules Guarantees Chase Termination

Acyclic Set of Rules Guarantees Chase Termination

A: $\mathrm{Request}(u)@x \rightarrow \mathrm{Payment}(u)@y,\ x \leq y$

B: $\mathrm{Request}(u)@x,\ \mathrm{Payment}(u)@y,\ x+5 \leq y \rightarrow \mathrm{Approval}(v)@z,\ z < x$

Acyclic Set of Rules Guarantees Chase Termination

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

Acyclic Set of Rules Guarantees Chase Termination

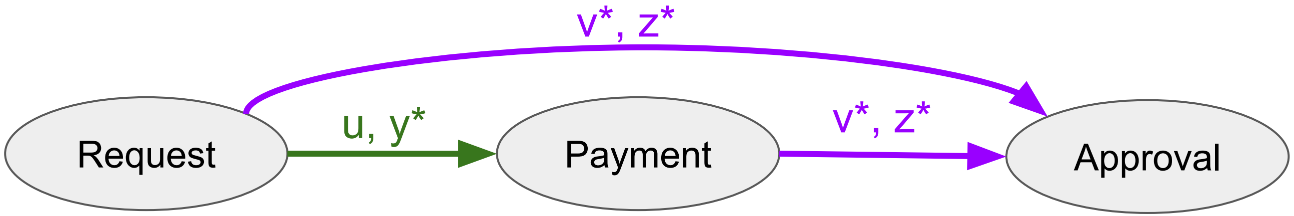
A: $\mathrm{Request}(u)@x \rightarrow \mathrm{Payment}(u)@y,\ x \leq y$

B: $\mathrm{Request}(u)@x,\ \mathrm{Payment}(u)@y,\ x+5 \leq y \rightarrow \mathrm{Approval}(v)@z,\ z < x$

Acyclic Set of Rules Guarantees Chase Termination

A: Request(u)@x → Payment(u)@y, x ≤ y

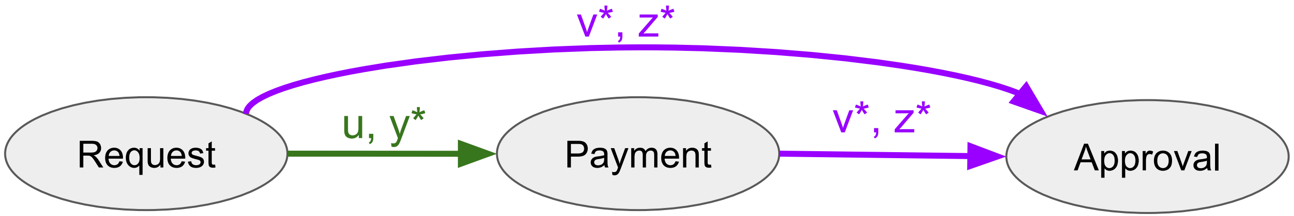
B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x



Acyclic Set of Rules Guarantees Chase Termination

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x



Chasing acyclic rules always terminates (Kolatis et al., PODS 2006)

Individual Rules: Match Assignments Before Deadlines

Individual Rules: Match Assignments Before Deadlines

$$\mathrm{Request}(u)@x,\ \mathrm{Payment}(u)@y,\ x+5 \leq y \quad \rightarrow \quad \mathrm{Approval}(v)@z,\ z < x$$

Individual Rules: Match Assignments Before Deadlines

Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

Body assignments				
id	u	x	y	gaps

Individual Rules: Match Assignments Before Deadlines

$Request(u)@x,\ Payment(u)@y,\ x+5 \leq y \quad \rightarrow \quad Approval(v)@z,\ z < x$

Body assignments				
id	u	x	y	gaps

Head assignments				
id	v	z	gaps	deadline

Individual Rules: Match Assignments Before Deadlines

Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

Body assignments				
id	u	x	y	gaps

Head assignments				
id	v	z	gaps	deadline

Extensions			
body	head	gaps	deadline

Individual Rules: Match Assignments Before Deadlines

Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

Body assignments				
id	u	x	y	gaps

Head assignments				
id	v	z	gaps	deadline

Extensions			
body	head	gaps	deadline

Individual Rules: Match Assignments Before Deadlines

Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

Body assignments				
id	u	x	y	gaps

Head assignments				
id	v	z	gaps	deadline

Extensions			
body	head	gaps	deadline

Approval(Bob)@9

Individual Rules: Match Assignments Before Deadlines

$\text{Request}(u)@x,\ \text{Payment}(u)@y,\ x+5 \le y \ \rightarrow\ \text{Approval}(v)@z,\ z < x$

Body assignments				
id	u	x	y	gaps

Head assignments				
id	v	z	gaps	deadline
β_2	Bob	9	$9 < x$	-

Extensions			
body	head	gaps	deadline

Approval(Bob)@9 →

Individual Rules: Match Assignments Before Deadlines

Request(u)@x, Payment(u)@y, $x+5 \leq y$ $\rightarrow$ Approval(v)@z, $z < x$

Body assignments				
id	u	x	y	gaps

Head assignments				
id	v	z	gaps	deadline
β_2	Bob	9	$9 < x$	-

Extensions			
body	head	gaps	deadline

Approval(Bob)@9 → Request(Alice)@10 →

Individual Rules: Match Assignments Before Deadlines

$\text{Request}(u)@x,\ \text{Payment}(u)@y,\ x+5 \leq y \ \rightarrow\ \text{Approval}(v)@z,\ z < x$

Body assignments				
id	u	x	y	gaps
α_1	Alice	10	-	$10 \leq y$

Head assignments				
id	v	z	gaps	deadline
β_2	Bob	9	$9 < x$	-

Extensions			
body	head	gaps	deadline

Approval(Bob)@9 → Request(Alice)@10 →

Individual Rules: Match Assignments Before Deadlines

$$\text{Request}(u)@x,\ \text{Payment}(u)@y,\ x+5 \leq y \ \rightarrow\ \text{Approval}(v)@z,\ z < x$$

Body assignments				
id	u	x	y	gaps
α_1	Alice	10	-	$10 \leq y$

Head assignments				
id	v	z	gaps	deadline
β_2	Bob	9	$9 < x$	-

Extensions			
body	head	gaps	deadline

Approval(Bob)@9 — Request(Alice)@10 — Payment(Alice)@16 →

Individual Rules: Match Assignments Before Deadlines

Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

Body assignments				
id	u	x	y	gaps
α_1	Alice	10	-	10 ≤ y
α_2	Alice	-	16	x ≤ 16

Head assignments				
id	v	z	gaps	deadline
β_2	Bob	9	9 < x	-

Extensions			
body	head	gaps	deadline

Approval(Bob)@9 → Request(Alice)@10 → Payment(Alice)@16 →

Individual Rules: Match Assignments Before Deadlines

$$\text{Request}(u)@x,\ \text{Payment}(u)@y,\ x+5 \leq y \ \rightarrow\ \text{Approval}(v)@z,\ z < x$$

Body assignments				
id	u	x	y	gaps
α_1	Alice	10	-	$10 \leq y$
α_2	Alice	-	16	$x \leq 16$
α_3	Alice	10	16	$10 \leq 16$

Head assignments				
id	v	z	gaps	deadline
β_2	Bob	9	$9 < x$	-

Extensions			
body	head	gaps	deadline

Approval(Bob)@9 → Request(Alice)@10 → Payment(Alice)@16 →

Individual Rules: Match Assignments Before Deadlines

$\text{Request}(u)@x,\ \text{Payment}(u)@y,\ x+5 \leq y \ \rightarrow \ \text{Approval}(v)@z,\ z < x$

Body assignments				
id	u	x	y	gaps
α_1	Alice	10	-	$10 \leq y$
α_2	Alice	-	16	$x \leq 16$
α_3	Alice	10	16	$10 \leq 16$

Head assignments				
id	v	z	gaps	deadline
β_2	Bob	9	$9 < x$	-

Extensions			
body	head	gaps	deadline
α_3	-	$z < 10$	10

Approval(Bob)@9 → Request(Alice)@10 → Payment(Alice)@16 →

Individual Rules: Match Assignments Before Deadlines

$\text{Request}(u)@x,\ \text{Payment}(u)@y,\ x+5 \leq y \ \rightarrow\ \text{Approval}(v)@z,\ z < x$

Body assignments				
id	u	x	y	gaps
α_1	Alice	10	-	$10 \leq y$
α_2	Alice	-	16	$x \leq 16$
α_3	Alice	10	16	$10 \leq 16$

Head assignments				
id	v	z	gaps	deadline
β_2	Bob	9	$9 < x$	-

Extensions			
body	head	gaps	deadline
α_3	-	$z < 10$	10
α_3	β_2	-	-

Approval(Bob)@9 → Request(Alice)@10 → Payment(Alice)@16 →

Expected Events Allow Violation Detection

Expected Events Allow Violation Detection

A: Request(u)@x → Payment(u)@y, $x \leq y$

B: Request(u)@x, Payment(u)@y, $x+5 \leq y$ → Approval(v)@z, $z < x$

Expected Events Allow Violation Detection

A: Request(u)@x → Payment(u)@y, $x \leq y$

B: Request(u)@x, Payment(u)@y, $x+5 \leq y$ → Approval(v)@z, $z < x$

B: Body assignments				
id	u	x	y	gaps

B: Head assignments				
id	v	z	gaps	deadline

B: Extensions			
body	head	gaps	deadline
α_3	-	$z < 10$	10

Expected Events Allow Violation Detection

A: Request(u)@x → Payment(u)@y, $x \leq y$

B: Request(u)@x, Payment(u)@y, $x+5 \leq y$ → Approval(v)@z, $z < x$

B: Body assignments					
id	u	x	y	gaps	Chased

B: Head assignments				
id	v	z	gaps	deadline

B: Extensions			
body	head	gaps	deadline
α_3	-	$z < 10$	10

Expected Events Allow Violation Detection

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

B: Body assignments					
id	u	x	y	gaps	Chased

B: Head assignments				
id	v	z	gaps	deadline

B: Extensions			
body	head	gaps	deadline
α_3	-	z < 10	10

Request(Alice)@10

Payment(Alice)@y', 10 ≤ y'

Expected Events Allow Violation Detection

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

B: Body assignments					
id	u	x	y	gaps	Chased
α_1	Alice	10	-	10 ≤ y	-
α_2	Alice	-	y'	x ≤ y'	-
α_3	Alice	10	y'	10 ≤ y'	-

B: Head assignments				
id	v	z	gaps	deadline

B: Extensions			
body	head	gaps	deadline
α_3	-	z < 10	10

Request(Alice)@10

Payment(Alice)@y', 10 ≤ y'

Expected Events Allow Violation Detection

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

B: Body assignments					
id	u	x	y	gaps	Chased
α_1	Alice	10	-	10 ≤ y	-
α_2	Alice	-	y'	x ≤ y'	-
α_3	Alice	10	y'	10 ≤ y'	*Yes*

B: Head assignments				
id	v	z	gaps	deadline

B: Extensions			
body	head	gaps	deadline
α_3	-	z < 10	10

Approval(Alice)@z', z' < 10
15 ≤ y'

Request(Alice)@10

Payment(Alice)@y', 10 ≤ y'

Expected Events Allow Violation Detection

A: Request(u)@x → Payment(u)@y, $x \leq y$

B: Request(u)@x, Payment(u)@y, $x+5 \leq y$ → Approval(v)@z, $z < x$

B: Body assignments					
id	u	x	y	gaps	Chased
α_1	Alice	10	-	$10 \leq y$	-
α_2	Alice	-	y'	$x \leq y'$	-
α_3	Alice	10	y'	$10 \leq y'$	Yes

B: Head assignments				
id	v	z	gaps	deadline
β_2	Bob	z'	-	-

B: Extensions			
body	head	gaps	deadline
α_3	-	$z < 10$	10

Approval(Alice)@z', $z' < 10$
$15 \leq y'$

Request(Alice)@10

Payment(Alice)@y', $10 \leq y'$

Expected Events Allow Violation Detection

A: Request(u)@x → Payment(u)@y, $x \leq y$

B: Request(u)@x, Payment(u)@y, $x+5 \leq y$ → Approval(v)@z, $z < x$

B: Body assignments					
id	u	x	y	gaps	Chased
α_1	Alice	10	-	$10 \leq y$	-
α_2	Alice	-	y'	$x \leq y'$	-
α_3	Alice	10	y'	$10 \leq y'$	*Yes*

B: Head assignments				
id	v	z	gaps	deadline
β_2	Bob	z'	-	-

B: Extensions			
body	head	gaps	deadline
α_3	-	$z < 10$	10
α_3	β_2	$z' < 10$ $15 \leq y'$	10

Approval(Alice)@z', $z' < 10$
$15 \leq y'$

Request(Alice)@10

Payment(Alice)@y', $10 \leq y'$

Expected Events Allow Violation Detection

A: Request(u)@x → Payment(u)@y, $x \leq y$

B: Request(u)@x, Payment(u)@y, $x+5 \leq y$ → Approval(v)@z, $z < x$

B: Body assignments					
id	u	x	y	gaps	Chased
α_1	Alice	10	-	$10 \leq y$	-
α_2	Alice	-	y'	$x \leq y'$	-
α_3	Alice	10	y'	$10 \leq y'$	Yes

B: Head assignments				
id	v	z	gaps	deadline
β_2	Bob	z'	-	-

B: Extensions			
body	head	gaps	deadline
α_3	-	$z < 10$	10
α_3	β_2	$z' < 10$ $15 \leq y'$	10

Approval(Alice)@z', $z' < 10$
$15 \leq y'$

Request(Alice)@10

Payment(Alice)@y', $10 \leq y'$

No violation at time *t* iff (when α_3 exists at *t*, it can match with β_2)

Expected Events Allow Violation Detection

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

B: Body assignments					
id	u	x	y	gaps	Chased
α_1	Alice	10	-	10 ≤ y	-
α_2	Alice	-	y'	x ≤ y'	-
α_3	Alice	10	y'	10 ≤ y'	*Yes*

B: Head assignments				
id	v	z	gaps	deadline
β_2	Bob	z'	-	-

B: Extensions			
body	head	gaps	deadline
α_3	-	z < 10	10
α_3	β_2	z' < 10 15 ≤ y'	10

Approval(Alice)@z', z' < 10
15 ≤ y'

Request(Alice)@10

Payment(Alice)@y', 10 ≤ y'

No violation at time *t* iff (when α_3 exists at *t*, it can match with β_2)

No violation at time *t* iff SAT((15 ≤ y' → z' < 10)

Expected Events Allow Violation Detection

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

B: Body assignments					
id	u	x	y	gaps	Chased
α_1	Alice	10	-	10 ≤ y	-
α_2	Alice	-	y'	x ≤ y'	-
α_3	Alice	10	y'	10 ≤ y'	*Yes*

B: Head assignments				
id	v	z	gaps	deadline
β_2	Bob	z'	-	-

B: Extensions			
body	head	gaps	deadline
α_3	-	z < 10	10
α_3	β_2	z' < 10 15 ≤ y'	10

Approval(Alice)@z', z' < 10
15 ≤ y'

Request(Alice)@10

Payment(Alice)@y', 10 ≤ y'

No violation at time *t* iff (when α_3 exists at *t*, it can match with β_2)

No violation at time *t* iff SAT((15 ≤ y' → z' < 10) ∧ (*t* < y') ∧ (*t* < z'))

Expected Events Allow Violation Detection

A: Request(u)@x → Payment(u)@y, x ≤ y

B: Request(u)@x, Payment(u)@y, x+5 ≤ y → Approval(v)@z, z < x

B: Body assignments					
id	u	x	y	gaps	Chased
α_1	Alice	10	-	10 ≤ y	-
α_2	Alice	-	y'	x ≤ y'	-
α_3	Alice	10	y'	10 ≤ y'	*Yes*

B: Head assignments				
id	v	z	gaps	deadline
β_2	Bob	z'	-	-

B: Extensions			
body	head	gaps	deadline
α_3	-	z < 10	10
α_3	β_2	z' < 10 15 ≤ y'	10

Approval(Alice)@z', z' < 10
15 ≤ y'

Request(Alice)@10

Payment(Alice)@y', 10 ≤ y'

No violation at time *t* iff (when α_3 exists at *t*, it can match with β_2)

No violation at time *t* iff SAT((15 ≤ y' → z' < 10) ∧ (*t* < y') ∧ (*t* < z'))

SAT while *t* < 15

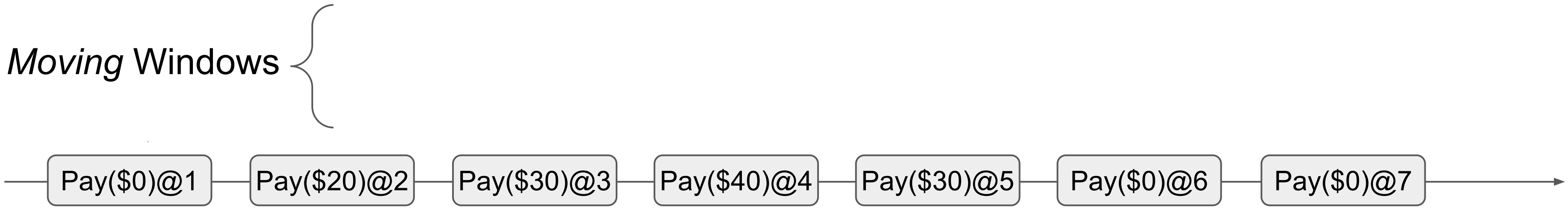
Outline

Rules

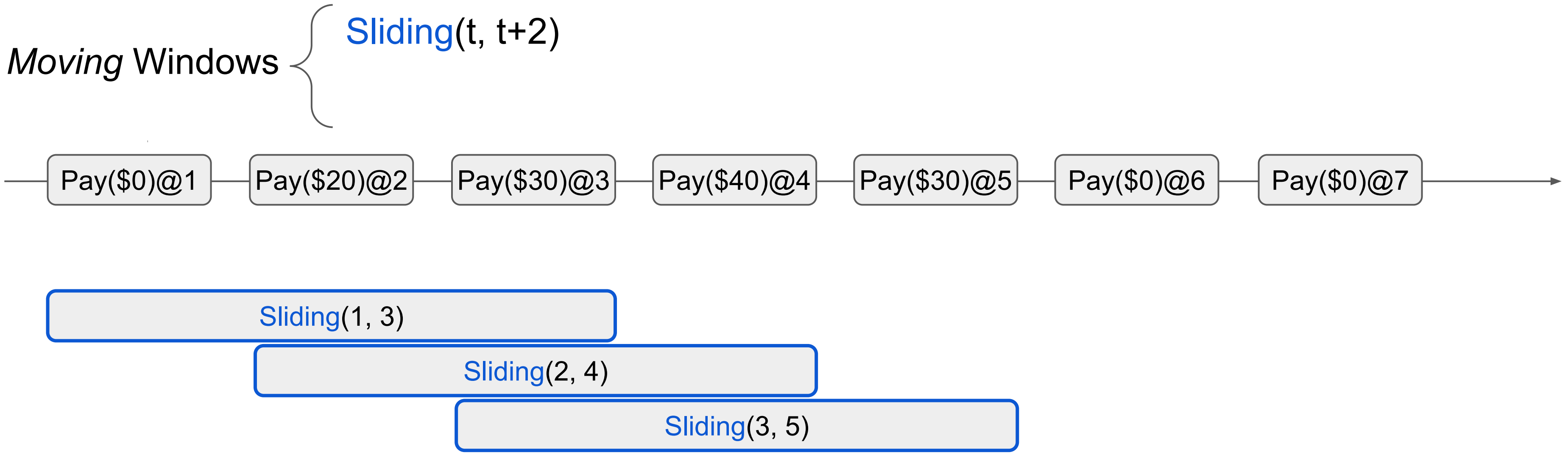
- **1.** without Data (Mackey & Su, Info. Sys., 2023), Comparison with Other Work
- with Data
  - **2.** Individual Rules (Mackey et al., TIME 2022)
  - Sets of Rules
    - Acyclic Sets
      - Chase Process (Mackey & Su, Info. & Comp., in revision, 2023)
      - **3. with Aggregation Functions on Time Windows**
    - **4.** Unrestricted Sets (Early Detection is Impossible)

Time Windows over Event Streams

Time Windows over Event Streams

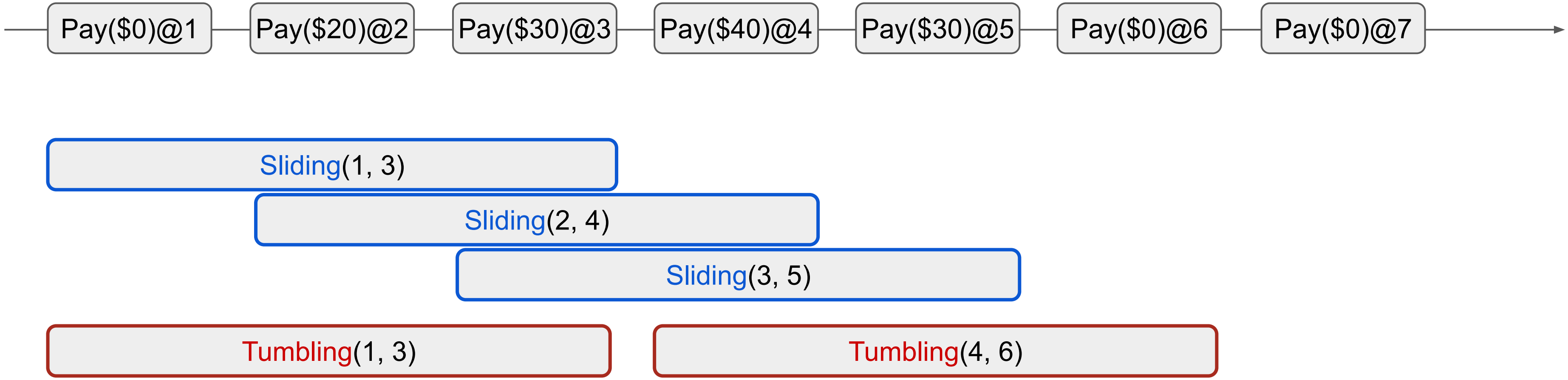


Time Windows over Event Streams



Time Windows over Event Streams

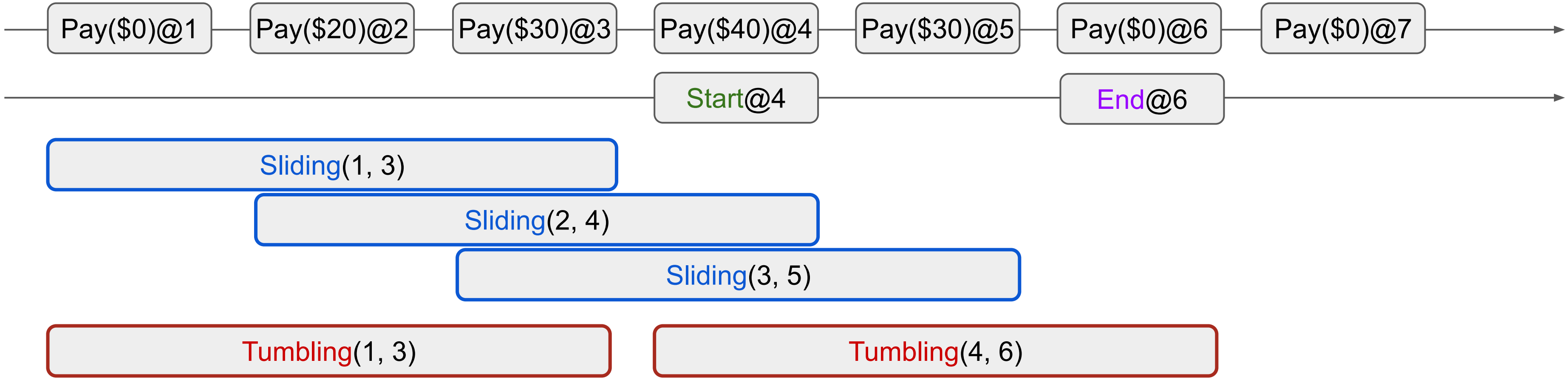
Moving Windows
- Sliding(t, t+2)
- Tumbling(t, t+2)



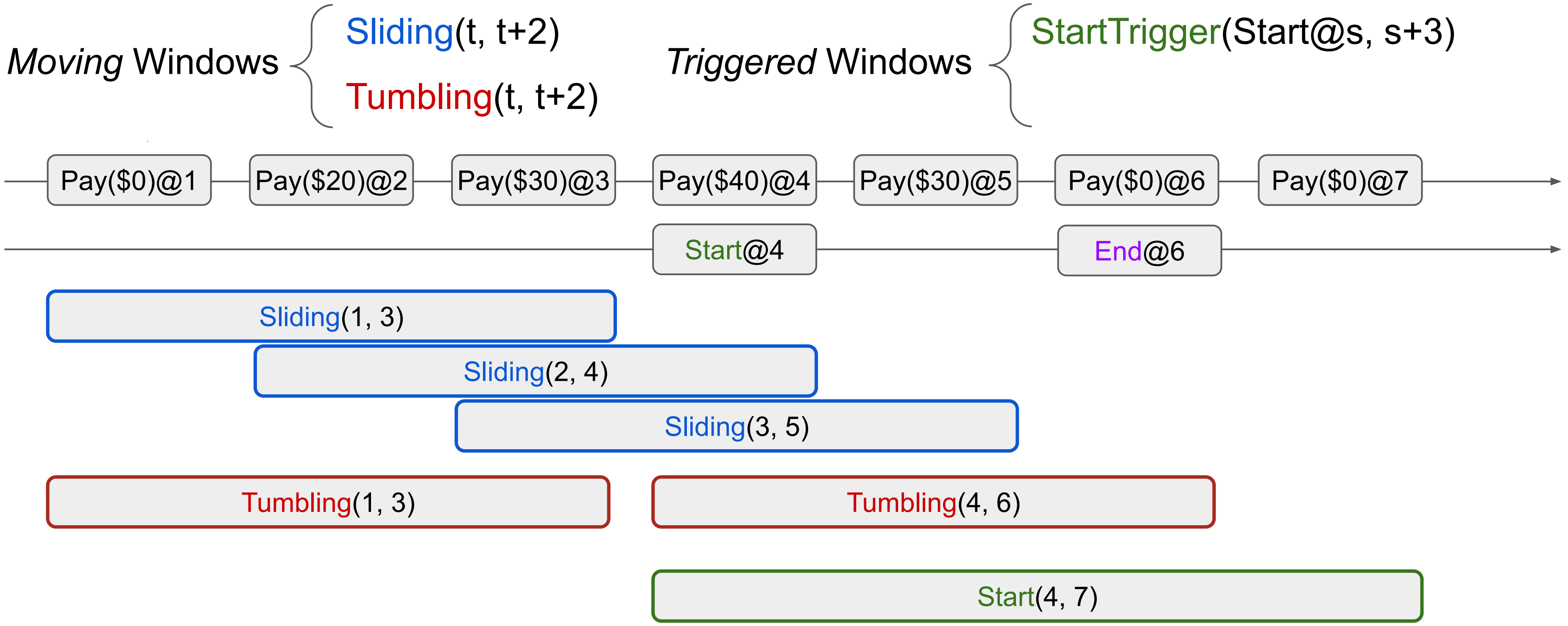
Time Windows over Event Streams

Moving Windows
Sliding(t, t+2)
Tumbling(t, t+2)

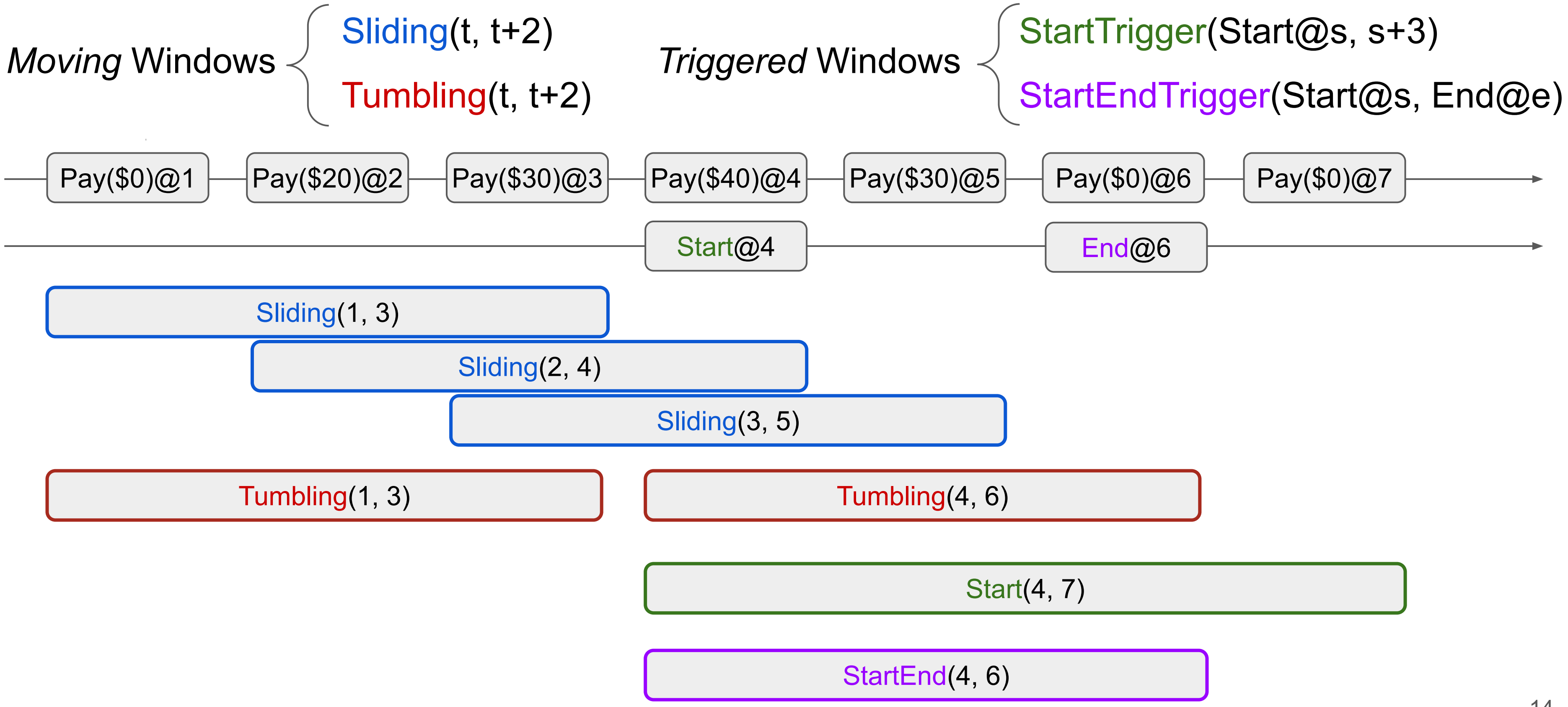
Triggered Windows



Time Windows over Event Streams



Time Windows over Event Streams



Aggregation Functions

Aggregation Functions

Aggregation functions: *sum*, *max*, *min*, *count, countu*

Aggregation Functions

Aggregation functions: *sum*, *max*, *min*, *count, countu*

SlidingSum(*s* = *sum*(a), t)@t+2

Aggregation Functions

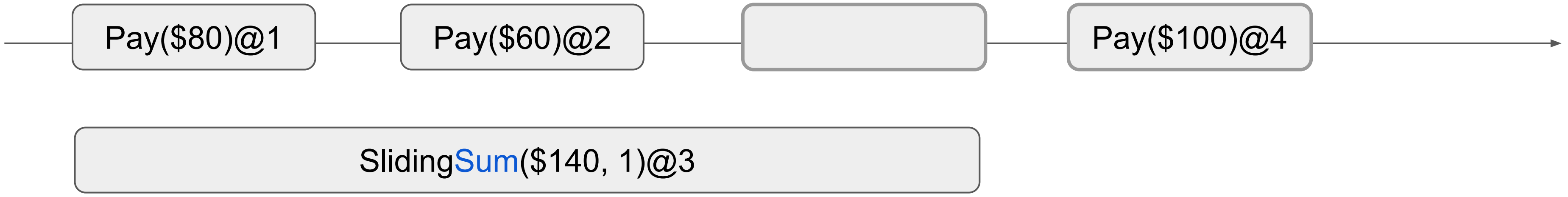
Aggregation functions: *sum*, *max*, *min*, *count, countu*

SlidingSum(*s* = *sum*(a), t)@t+2

Aggregation Functions

Aggregation functions: *sum*, *max*, *min*, *count, countu*

SlidingSum(*s* = *sum*(a), t)@t+2

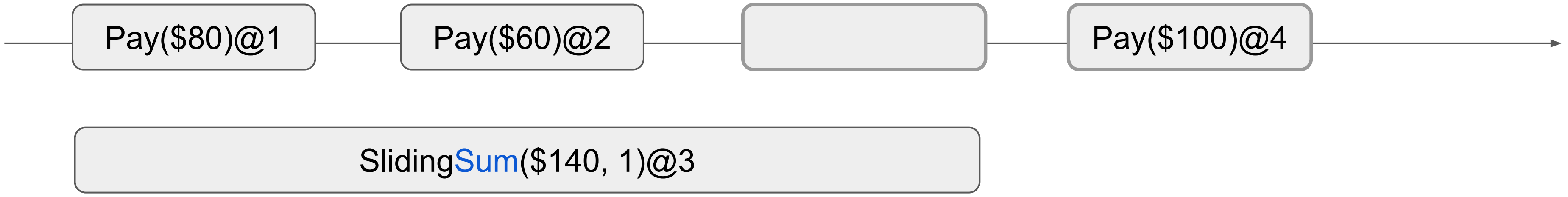


Aggregation Functions

Aggregation functions: *sum*, *max*, *min*, *count, countu*

SlidingSum(*s* = *sum*(a), t)@t+2

SlidingMax(*m* = *max*(a), t)@t+2



Aggregation Functions

Aggregation functions: *sum*, *max*, *min*, *count, countu*

SlidingSum(*s* = *sum*(a), t)@t+2

SlidingMax(*m* = *max*(a), t)@t+2

Pay($80)@1 Pay($60)@2 Pay($100)@4

SlidingSum($140, 1)@3

SlidingMax($80, 1)@3

Chasing Windows with Presburger Arithmetic

Chasing Windows with Presburger Arithmetic

For each (open) window and aggregation function (*sum*, *max*, *min*, *count, countu*), there is an equivalent Presburger arithmetic constraint.

Chasing Windows with Presburger Arithmetic

For each (open) window and aggregation function (*sum*, *max*, *min*, *count, countu*), there is an equivalent Presburger arithmetic constraint.

Chasing Windows with Presburger Arithmetic

For each (open) window and aggregation function (*sum*, *max*, *min*, *count, countu*), there is an equivalent Presburger arithmetic constraint.

Chasing Windows with Presburger Arithmetic

For each (open) window and aggregation function (*sum*, *max*, *min*, *count, countu*), there is an equivalent Presburger arithmetic constraint.

SlidingSum(*s* = sum(a), t)@t+2

Chasing Windows with Presburger Arithmetic

For each (open) window and aggregation function (*sum*, *max*, *min*, *count, countu*), there is an equivalent Presburger arithmetic constraint.

SlidingSum(s = sum(a), t)@t+2

Pay($80)@1 — Pay($60)@2 — Pay(a)@3 →

SlidingSum(s', 1)@3, $s' = 80 + 60 + a$

Chasing Windows with Presburger Arithmetic

For each (open) window and aggregation function (*sum*, *max*, *min*, *count, countu*), there is an equivalent Presburger arithmetic constraint.

SlidingSum(s = sum(a), t)@t+2

SlidingMax(m = max(a), t)@t+2

Pay($80)@1 — Pay($60)@2 — Pay(a)@3 →

SlidingSum(s', 1)@3, $s' = 80 + 60 + a$

Chasing Windows with Presburger Arithmetic

For each (open) window and aggregation function (*sum*, *max*, *min*, *count, countu*), there is an equivalent Presburger arithmetic constraint.

SlidingSum(s = sum(a), t)@t+2

SlidingMax(m = max(a), t)@t+2

Pay($80)@1 — Pay($60)@2 — Pay(a)@3

SlidingSum(s', 1)@3, $s' = 80 + 60 + a$

SlidingMax(m', 1)@3, $((m' = 80) \lor (m' = 60) \lor (m' = a)) \land ((80 \leq m') \land (60 \leq m') \land (a \leq m'))$

Detecting Violations

Detecting Violations

$$\begin{cases} \text{SlidingSum}(\ s = sum(\text{a}),\ \text{t})@\text{t+2} \rightarrow s \leq 200 \\ \text{SlidingMax}(\ m = max(\text{a}),\ \text{t})@\text{t+2} \rightarrow m \geq 100 \end{cases}$$

Detecting Violations

$$\begin{cases} \text{SlidingSum}(\ s = sum(\text{a}),\ \text{t})@\text{t}+2 \ \rightarrow s \leq 200 \\ \text{SlidingMax}(\ m = max(\text{a}),\ \text{t})@\text{t}+2 \rightarrow m \geq 100 \end{cases}$$

Pay($80)@1 — Pay($60)@2 — Pay(*a*)@3

Detecting Violations

$$\begin{cases} \text{SlidingSum}(\ s = sum(\text{a}),\ \text{t})@\text{t+2} \ \rightarrow s \le 200 \\ \text{SlidingMax}(\ m = max(\text{a}),\ \text{t})@\text{t+2} \rightarrow m \ge 100 \end{cases}$$

Pay(\$80)@1 — Pay(\$60)@2 — Pay(a)@3

SlidingSum(s', 1)@3, $s' = 80 + 60 + a$

SlidingMax(m', 1)@3, $((m' = 80) \lor (m' = 60) \lor (m' = a))$
$\land$
$((80 \le m') \land (60 \le m') \land (a \le m'))$

Detecting Violations

Theorem. The earliest violation for an acyclic set of rules with aggregation can be computed.

$$\begin{cases} \text{SlidingSum}(\ s = sum(\text{a}),\ \text{t})@\text{t}+2 \ \rightarrow s \le 200 \\ \text{SlidingMax}(\ m = max(\text{a}),\ \text{t})@\text{t}+2 \rightarrow m \ge 100 \end{cases}$$

Pay(\$80)@1 — Pay(\$60)@2 — Pay(a)@3

SlidingSum(s', 1)@3, $s' = 80 + 60 + a$

SlidingMax(m', 1)@3, $((m' = 80) \lor (m' = 60) \lor (m' = a))$
$\land$
$((80 \le m') \land (60 \le m') \land (a \le m'))$

Detecting Violations

Theorem. The earliest violation for an acyclic set of rules with aggregation can be computed.

SlidingSum($s = sum(a)$, t)@t+2 $\rightarrow s \leq 200$

SlidingMax($m = max(a)$, t)@t+2 $\rightarrow m \geq 100$

SlidingSum Head assignments			
id	t	s	gaps
α_1	1	s'	$s' = 80 + 60 + a$ $s' \leq 200$

Pay($80)@1 — Pay($60)@2 — Pay(a)@3

SlidingSum(s', 1)@3, $s' = 80 + 60 + a$

SlidingMax(m', 1)@3, $((m' = 80) \vee (m' = 60) \vee (m' = a))$
$\wedge$
$((80 \leq m') \wedge (60 \leq m') \wedge (a \leq m'))$

Detecting Violations

Theorem. The earliest violation for an acyclic set of rules with aggregation can be computed.

$$\begin{cases} \text{SlidingSum}(\ s = sum(\text{a}),\ \text{t})@\text{t+2} \ \rightarrow s \leq 200 \\ \text{SlidingMax}(\ m = max(\text{a}),\ \text{t})@\text{t+2} \rightarrow m \geq 100 \end{cases}$$

Pay($80)@1 — Pay($60)@2 — Pay(a)@3

SlidingSum(s', 1)@3, $s' = 80 + 60 + a$

SlidingMax(m', 1)@3, $((m' = 80) \lor (m' = 60) \lor (m' = a)) \land ((80 \leq m') \land (60 \leq m') \land (a \leq m'))$

SlidingSum Head assignments			
id	t	s	gaps
α_1	1	s'	$s' = 80 + 60 + a$ $s' \leq 200$

SlidingSum Head assignments			
id	t	m	gaps
β_1	1	m'	$((m' = 80) \lor (m' = 60) \lor (m' = a))$ $\land ((80 \leq m') \land (60 \leq m') \land (a \leq m'))$, $m \geq 100$

Detecting Violations

Theorem. The earliest violation for an acyclic set of rules with aggregation can be computed.

$$\begin{cases} \text{SlidingSum}(\, s = sum(\text{a}),\ \text{t})@\text{t+2} \ \rightarrow s \leq 200 \\ \text{SlidingMax}(\, m = max(\text{a}),\ \text{t})@\text{t+2} \rightarrow m \geq 100 \end{cases}$$

Pay($80)@1 — Pay($60)@2 — Pay(a)@3

SlidingSum(s', 1)@3, $s' = 80 + 60 + a$

SlidingMax(m', 1)@3, $((m' = 80) \lor (m' = 60) \lor (m' = a))$
$\land$
$((80 \leq m') \land (60 \leq m') \land (a \leq m'))$

SlidingSum Head assignments			
id	t	s	gaps
α_1	1	s'	$s' = 80 + 60 + a$ $s' \leq 200$

$a \leq 60$

SlidingSum Head assignments			
id	t	m	gaps
β_1	1	m'	$((m' = 80) \lor (m' = 60) \lor (m' = a))$ $\land ((80 \leq m') \land (60 \leq m') \land (a \leq m'))$, $m \geq 100$

Detecting Violations

Theorem. The earliest violation for an acyclic set of rules with aggregation can be computed.

$$\begin{cases} \text{SlidingSum}(\, s = sum(\text{a}), \text{t})@\text{t+2} \rightarrow s \leq 200 \\ \text{SlidingMax}(\, m = max(\text{a}), \text{t})@\text{t+2} \rightarrow m \geq 100 \end{cases}$$

Pay($80)@1 — Pay($60)@2 — Pay(a)@3

SlidingSum(s', 1)@3, $s' = 80 + 60 + a$

SlidingMax(m', 1)@3, $((m' = 80) \lor (m' = 60) \lor (m' = a)) \land ((80 \leq m') \land (60 \leq m') \land (a \leq m'))$

SlidingSum Head assignments			
id	t	s	gaps
α_1	1	s'	$s' = 80 + 60 + a$ $s' \leq 200$

$a \leq 60$

SlidingSum Head assignments			
id	t	m	gaps
β_1	1	m'	$((m' = 80) \lor (m' = 60) \lor (m' = a))$ $\land ((80 \leq m') \land$ [illegible] $m')), m \geq 100$

$a \geq 100$

Detecting Violations

Theorem. The earliest violation for an acyclic set of rules with aggregation can be computed.

$$\left\{\begin{array}{l} \mathrm{SlidingSum}(\ s = sum(\mathrm{a}),\ \mathrm{t})@\mathrm{t}{+}2 \ \rightarrow s \le 200 \\ \mathrm{SlidingMax}(\ m = max(\mathrm{a}),\ \mathrm{t})@\mathrm{t}{+}2 \rightarrow m \ge 100 \end{array}\right.$$

$\mathrm{Pay}(\$80)@1$ — $\mathrm{Pay}(\$60)@2$ — $\mathrm{Pay}(a)@3$

$\mathrm{SlidingSum}(s', 1)@3,\ s' = 80 + 60 + a$

$\mathrm{SlidingMax}(m', 1)@3,\ ((m' = 80) \vee (m' = 60) \vee (m' = a)) \wedge ((80 \le m') \wedge (60 \le m') \wedge (a \le m'))$

SlidingSum Head assignments

id	t	s	gaps
α_1	1	s'	$s' = 80 + 60 + a$ $s' \le 200$

$a \le 60$

SlidingSum Head assignments

id	t	m	gaps
β_1	1	m'	$((m' = 80) \vee (m' = 60) \vee (m' = a))$ $\wedge ((80 \le m') \wedge$ [illegible] $m')), m \ge 100$

$a \ge 100$

No violation at $t = 2$ iff
SAT((true $\rightarrow \alpha_1$) $\wedge$ (true $\rightarrow \beta_1$))

Outline

- Rules
  - **1.** without Data (Mackey & Su, Info. Sys., 2023), Comparison with Other Work
  - with Data
    - **2.** Individual Rules (Mackey et al., TIME 2022)
    - Sets of Rules
      - Acyclic Sets
        - Chase Process

          (Mackey & Su, Info. & Comp., in revision, 2023)
        - **3.** with Aggregation Functions on Time Windows
      - **4.** **Unrestricted Sets (Early Detection is Impossible)**

Early Violation Detection is Impossible for an Arbitrary Set of Rules

Early Violation Detection is Impossible for an Arbitrary Set of Rules

Theorem. Early violation detection for a set of rules is impossible.

Early Violation Detection is Impossible for an Arbitrary Set of Rules

Theorem. Early violation detection for a set of rules is impossible.

Proof Idea: We introduce *finite satisfiability* for a set of rules,

Early Violation Detection is Impossible for an Arbitrary Set of Rules

Theorem. Early violation detection for a set of rules is impossible.

Proof Idea: We introduce *finite satisfiability* for a set of rules,

- Finite Satisfiability: given a set of rules R, is there a finite event stream that satisfies R?

Early Violation Detection is Impossible for an Arbitrary Set of Rules

Theorem. Early violation detection for a set of rules is impossible.

Proof Idea: We introduce *finite satisfiability* for a set of rules,

- Finite Satisfiability: given a set of rules R, is there a finite event stream that satisfies R?

which reduces to early violation detection,

Early Violation Detection is Impossible for an Arbitrary Set of Rules

Theorem. Early violation detection for a set of rules is impossible.

Proof Idea: We introduce *finite satisfiability* for a set of rules,

- Finite Satisfiability: given a set of rules R, is there a finite event stream that satisfies R?

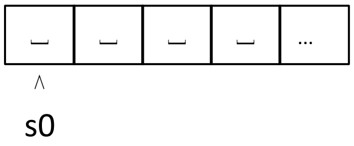
which reduces to early violation detection,

and we show finite satisfiability is undecidable by a reduction from the empty-tape Turing machine halting problem.

Encoding configurations of TM with **Config** events

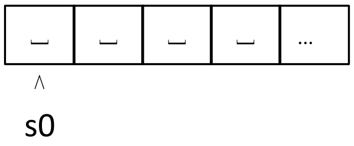
Encoding configurations of TM with **Config** events

configurations



Encoding configurations of TM with **Config** events

configurations



Config(0, #, -)

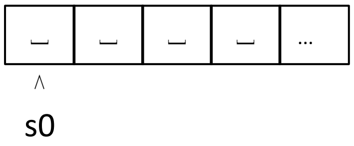
Config(1, ␣, s0)

Config(2, #, -)

Encoding configurations of TM with **Config** events

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-

configurations



Config(0, #, -)

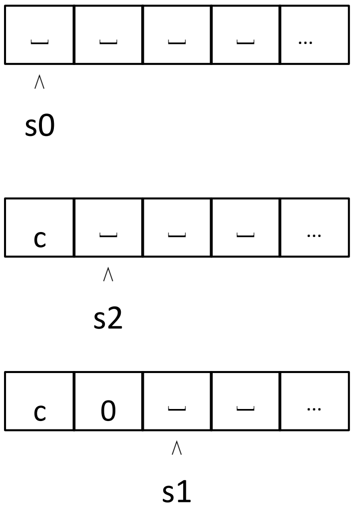
Config(1, ␣, s0)

Config(2, #, -)

Encoding configurations of TM with **Config** events

Config		
index	tape	state
0	#	-
1	⎵	s0
2	#	-

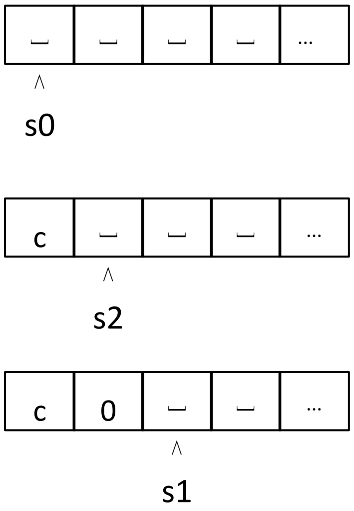
configurations



Encoding configurations of TM with **Config** events

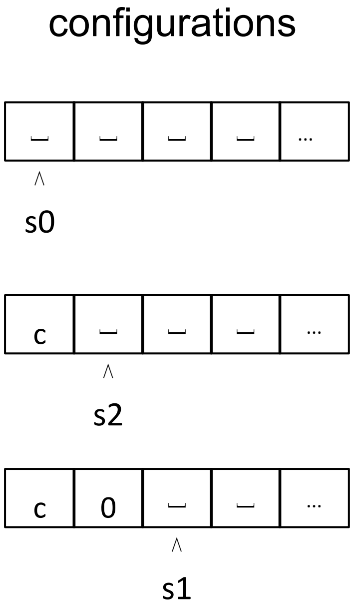
Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-
3	c	-
4	␣	s2
5	#	-

configurations



Encoding configurations of TM with **Config** events

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-
3	c	-
4	␣	s2
5	#	-
6	c	-
7	0	-
8	␣	s1
9	#	-



Encoding initial empty-tape configuration

Config		
index	tape	state

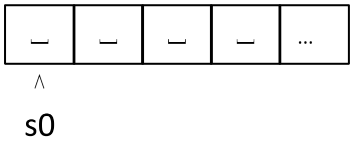
␣	␣	␣	␣	…

^

s0

Encoding initial empty-tape configuration

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-



Encoding initial empty-tape configuration

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-

␣	␣	␣	␣	…

^

s0

If M starts in s0, then R_M has:

true

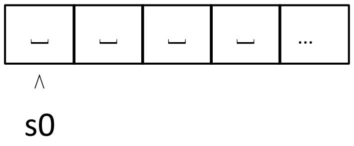
→

Config(0, #, -), Config(1, ␣, s0), Config(2, #, -)

Encoding initial empty-tape configuration

Next	
index	next
0	2
1	3
2	5

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-



If M starts in s0, then R_M has:

true

→

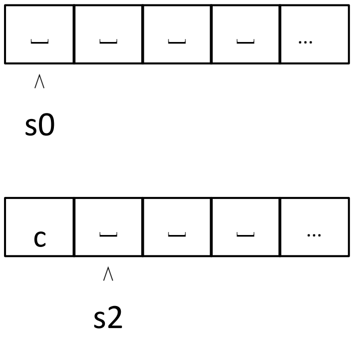
Config(0, #, -), Config(1, ␣, s0), Config(2, #, -)

Next(0, 2), Next(1, 3), Next(2, 5),

Encoding transitions of TM with **Config**, **Next**, and rules

Next	
index	next
0	2
1	3
2	5

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-

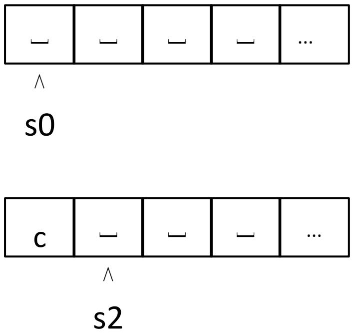


If M has δ(s0, ␣) = (c, s2, R), then R_M has:

Encoding transitions of TM with **Config**, **Next**, and rules

Next	
index	next
0	2
1	3
2	5
3	6
4	7
5	9

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-
3	c	-
4	␣	s2
5	#	-



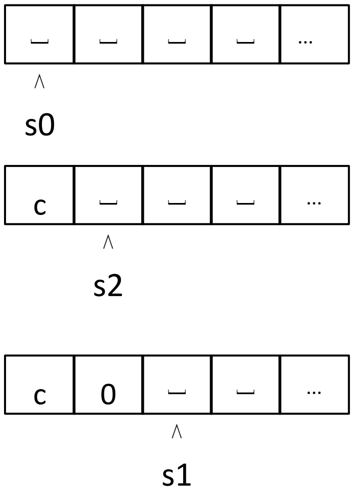
If M has $\boldsymbol{\delta}$(s0, ␣) = (c, s2, R), then R_M has:

Next(x-1, y-1), Config(x-1, #, -), Config(x, ␣, s0),
Config(x+1, #, -)
→
Next(x+2, y+3), Config(y-1, #, -), Config(y, c, -),
Config(y+1, ␣, s2), Config(x+2, #, -)

Encoding transitions of TM with **Config**, **Next**, and rules

Next	
index	next
0	2
1	3
2	5
3	6
4	7
5	9
6	10
7	11
8	12
9	13

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-
3	c	-
4	␣	s2
5	#	-
6	c	-
7	0	-
8	␣	s1
9	#	-



If M has δ(s2, ␣) = (0, s1, R), then R_M has:

Next(x-1, y-1), Config(x-1, #, -), Config(x, ␣, s2), Config(x+1, #, -)

→

Next(x+2, y+3), Config(y-1, #, -), Config(y, 0, -), Config(y+1, ␣, s1), Config(x+2, #, -)

Detect non-valid computations with **Error** rules

Detect non-valid computations with **Error** rules

Next	
index	next
0	2
1	3
2	5
3	6
4	7
5	9
6	10
7	11
8	12
9	14

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-
3	c	-
3	␣	s2
4	␣	-
5	#	-
6	c	-
7	0	-
8	␣	s1

Detect non-valid computations with **Error** rules

Next	
index	next
0	2
1	3
2	5
3	6
4	7
5	9
6	10
7	11
8	12
9	14

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-
3	c	-
3	␣	s2
4	␣	-
5	#	-
6	c	-
7	0	-
8	␣	s1

Detect non-valid computations with **Error** rules

Next	
index	next
0	2
1	3
2	5
3	6
4	7
5	9
6	10
7	11
8	12
9	14

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-
3	c	-
3	␣	s2
4	␣	-
5	#	-
6	c	-
7	0	-
8	␣	s1

Error
x

Detect non-valid computations with **Error** rules

Next	
index	next
0	2
1	3
2	5
3	6
4	7
5	9
6	10
7	11
8	12
9	14

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-
3	c	-
3	␣	s2
4	␣	-
5	#	-
6	c	-
7	0	-
8	␣	s1

Error
x
0

Don't allow malformed configurations:

Config(x, a, s), Config(x, b, s'), a ≠ b

→

Error(0)

Detect non-valid computations with **Error** rules

Next	
index	next
0	2
1	3
2	5
3	6
4	7
5	9
6	10
7	11
8	12
9	14

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-
3	c	-
3	␣	s2
4	␣	-
5	#	-
6	c	-
7	0	-
8	␣	s1

Error
x
0

Don't allow malformed configurations:

Config(x, a, s), Config(x, b, s'), a ≠ b

→

Error(0)

Propagate Errors infinitely:

Error(x)

→

Error(x+1)

Detect non-valid computations with **Error** rules

Next	
index	next
0	2
1	3
2	5
3	6
4	7
5	9
6	10
7	11
8	12
9	14

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-
3	c	-
3	␣	s2
4	␣	-
5	#	-
6	c	-
7	0	-
8	␣	s1

Error
x
0
1
2
3
...

Don't allow malformed configurations:

Config(x, a, s), Config(x, b, s'), a ≠ b

→

Error(0)

Propagate Errors infinitely:

Error(x)

→

Error(x+1)

Detect non-valid computations with **Error** rules

Next	
index	next
0	2
1	3
2	5
3	6
4	7
5	9
6	10
7	11
8	12
9	14

Config		
index	tape	state
0	#	-
1	␣	s0
2	#	-
3	c	-
3	␣	s2
4	␣	-
5	#	-
6	c	-
7	0	-
8	␣	s1

Error
x
0
1
2
3
...

Don't allow malformed configurations:

Config(x, a, s), Config(x, b, s'), a ≠ b

→

Error(0)

Propagate Errors infinitely:

Error(x)

→

Error(x+1)

For a Turing machine M, the set R_M is finitely satisfiable *iff* M halts on empty tape.

Outline

Rules

1. without Data (Mackey & Su, Info. Sys., 2023), Comparison with Other Work

with Data

2. Individual Rules (Mackey et al., TIME 2022)

Sets of Rules

Acyclic Sets

Chase Process

(Mackey & Su, Info. & Comp., in revision, 2023)

3. with Aggregation Functions on Time Windows

4. Unrestricted Sets (Early Detection is Impossible)

Thesis Organization

- Rules
  - Chapter 3. without Data (Mackey & Su, Info. Sys., 2023), Comparison with Other Work
  - with Data
    - Chapter 4. Individual Rules (Mackey et al., TIME 2022)
    - Sets of Rules
      - Acyclic Sets
        - Chase Process (Mackey & Su, Info. & Comp., in revision, 2023)
        - Chapter 5. with Aggregation Functions on Time Windows
      - Chapter 6. Unrestricted Sets (Early Detection is Impossible)

Summary of Contributions to Early Violation Detection

Summary of Contributions to Early Violation Detection

- We improve the size complexity of translations from two subclases of dataless rules to LTL.

Summary of Contributions to Early Violation Detection

- We improve the size complexity of translations from two subclases of dataless rules to LTL.
- We provide algorithms to detect violations of an acyclic set of rules at the earliest possible time, including rules with aggregation functions.

Summary of Contributions to Early Violation Detection

- We improve the size complexity of translations from two subclases of dataless rules to LTL.
- We provide algorithms to detect violations of an acyclic set of rules at the earliest possible time, including rules with aggregation functions.
- We show early violation detection for an arbitrary set of rules is impossible.

Summary of Contributions to Early Violation Detection

- We improve the size complexity of translations from two subclases of dataless rules to LTL.
- We provide algorithms to detect violations of an acyclic set of rules at the earliest possible time, including rules with aggregation functions.
- We show early violation detection for an arbitrary set of rules is impossible.

Future Directions

Summary of Contributions to Early Violation Detection

- We improve the size complexity of translations from two subclases of dataless rules to LTL.
- We provide algorithms to detect violations of an acyclic set of rules at the earliest possible time, including rules with aggregation functions.
- We show early violation detection for an arbitrary set of rules is impossible.

Future Directions

- Can violations of more complex time constraints be detected early?

Summary of Contributions to Early Violation Detection

- We improve the size complexity of translations from two subclases of dataless rules to LTL.
- We provide algorithms to detect violations of an acyclic set of rules at the earliest possible time, including rules with aggregation functions.
- We show early violation detection for an arbitrary set of rules is impossible.

Future Directions

- Can violations of more complex time constraints be detected early?
- Do richer sets of rules have (efficient) algorithms, e.g., negation, disjunction?

Thank you! Questions?