VeriMon

A Formally Verified Monitoring Tool

Dmitriy Traytel
traytel@di.ku.dk

UNIVERSITY OF COPENHAGEN

VeriMon

A Formally Verified Monitoring Tool

Dmitriy Traytel

traytel@di.ku.dk

joint work with

David Basin, Thibault Dardinier, Nico Hauser, Lukas Heimes, Jonathan Julián Huerta y Munive, Nicolas Kaletsch, Srđan Krstić, Emanuele Marsicano, Martin Raszyk, Joshua Schneider, Dawit Legesse Tirore, Sheila Zingg

UNIVERSITY OF COPENHAGEN

ETH zürich

DFINITY

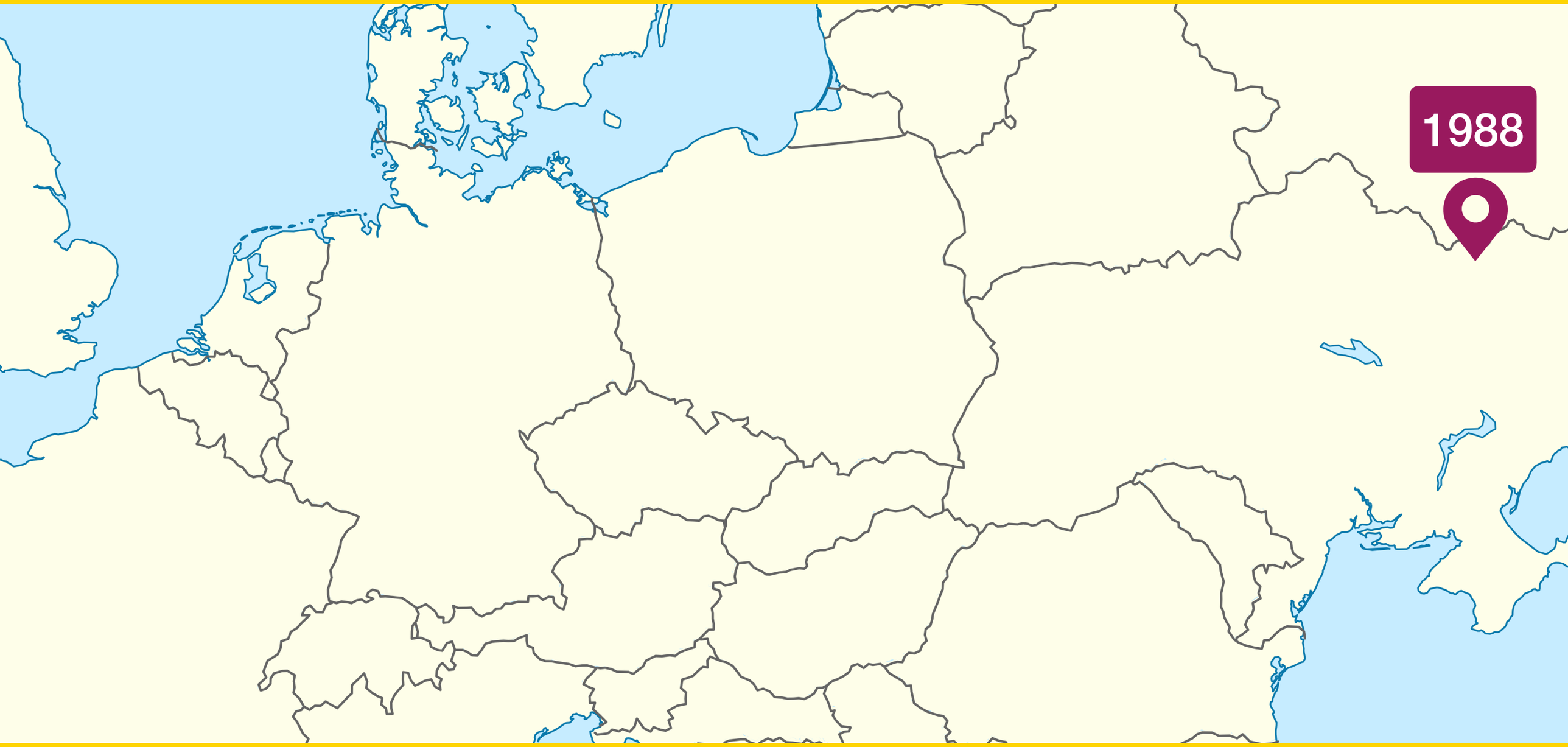
IT UNIVERSITY OF CPH

1 Monitoring

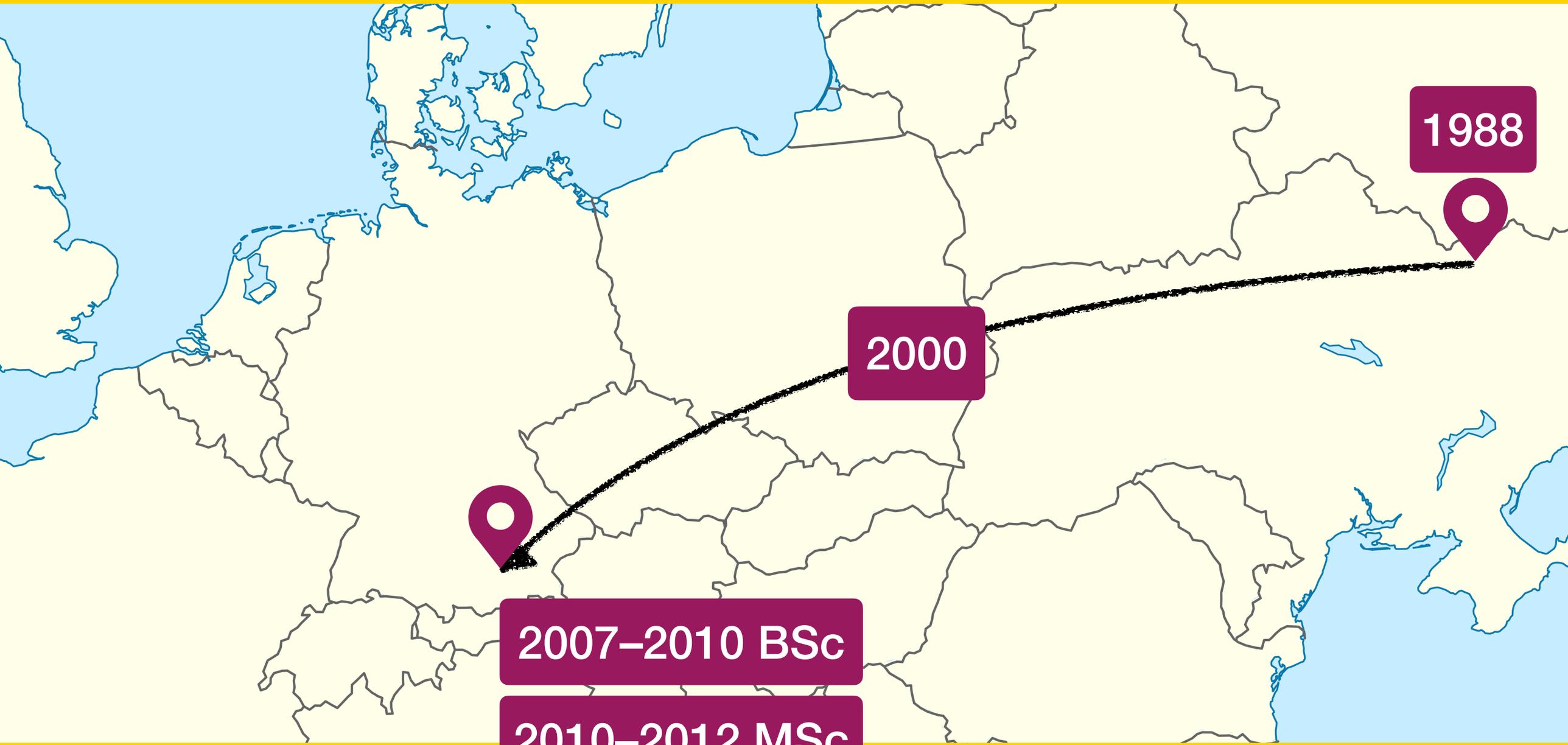
2 Formally Verified in Isabelle/HOL

3 VeriMon

About Myself

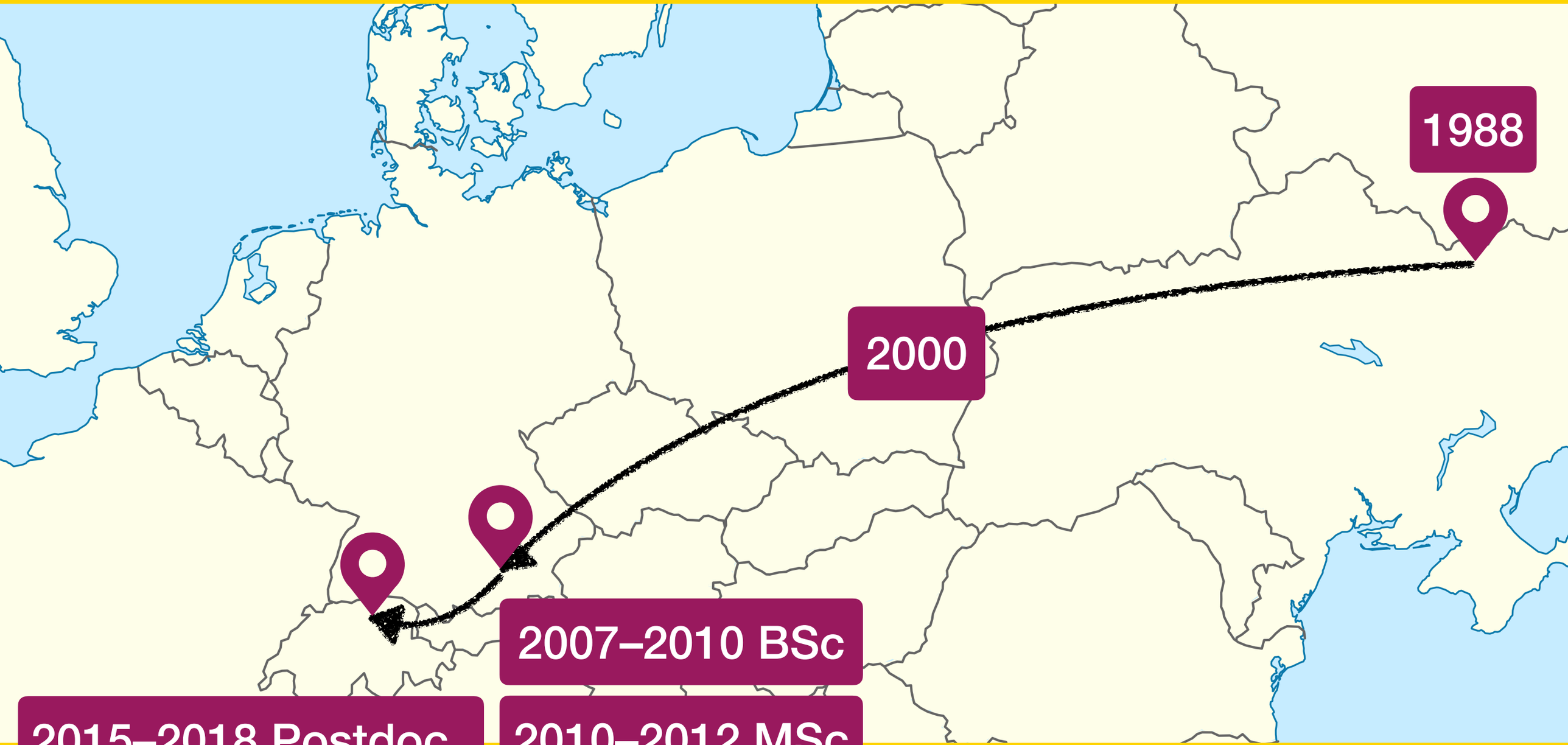


About Myself



Technische Universität München

About Myself

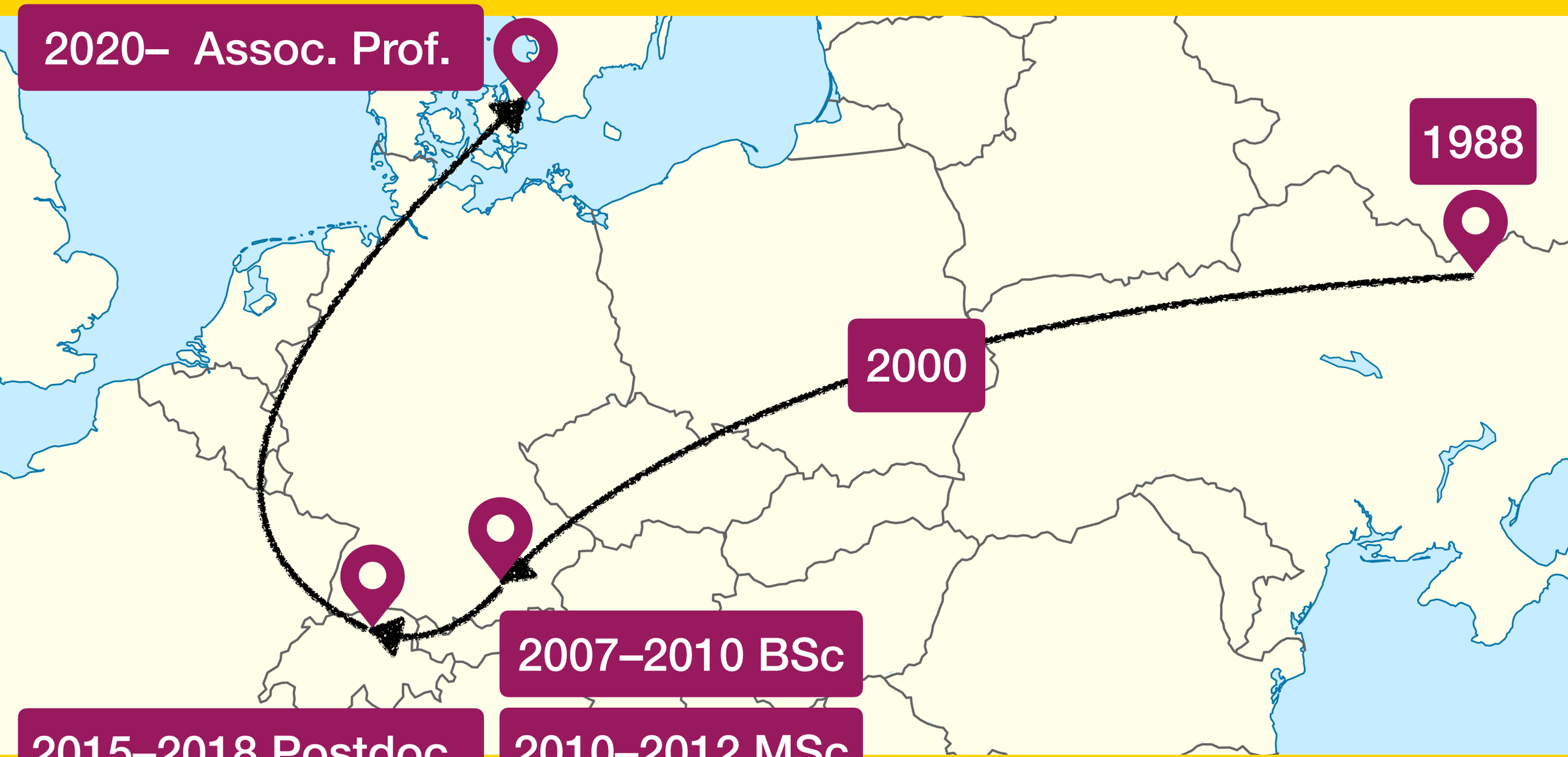


ETH zürich

Technische Universität München

UNIVERSITY OF COPENHAGEN

About Myself



ETH zürich

Technische Universität München

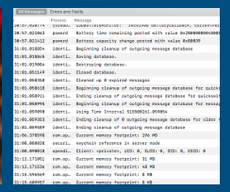
My Research

My Research

Analyse/Monitor
Running Systems

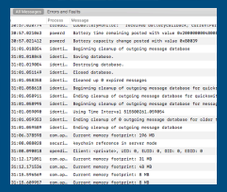
My Research

Analyse/Monitor Running Systems



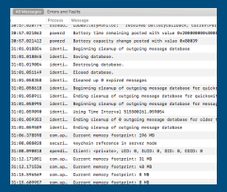
My Research

Analyse/Monitor Running Systems



My Research

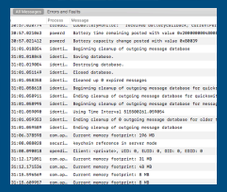
Analyse/Monitor Running Systems



Long-running SSH sessions must not last longer than 24 hours.

My Research

Analyse/Monitor Running Systems

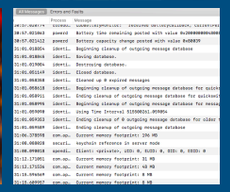


$$\Box \forall c.\ \forall s.\ ssh_login(c,\ s) \wedge \big((\Diamond_{[1min,20min]}\ net(c)) \wedge \Box_{[0,1d]}\ (\blacksquare_{=0}\ net(c) \rightarrow \Diamond_{[1min,20min]}\ net(c))\big) \rightarrow \Diamond_{[0,1d)}\ \blacklozenge_{=0}\ ssh_logout(c,\ s)$$

Long-running SSH sessions must not last longer than 24 hours.

My Research

Analyse/Monitor Running Systems

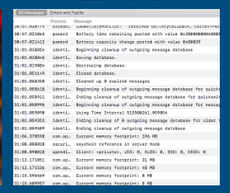


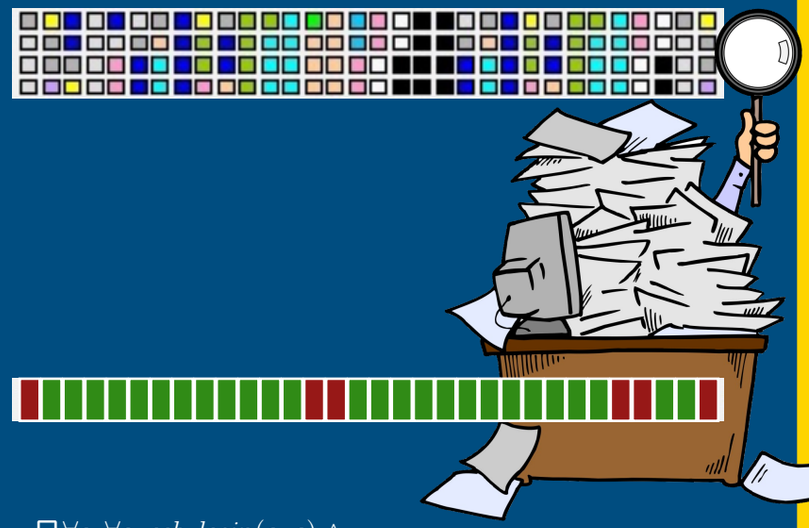
$$\Box \forall c.\ \forall s.\ ssh_login(c,\ s) \wedge \big((\Diamond_{[1min,20min]}\ net(c)) \wedge \Box_{[0,1d]}\ (\blacksquare_{=0}\ net(c) \rightarrow \Diamond_{[1min,20min]}\ net(c))\big) \rightarrow \Diamond_{[0,1d)}\ \blacklozenge_{=0}\ ssh_logout(c,\ s)$$

Long-running SSH sessions must not last longer than 24 hours.

My Research

Analyse/Monitor Running Systems





$$\Box \forall c.\ \forall s.\ ssh_login(c,\ s) \wedge \big((\Diamond_{[1min,20min]}\ net(c)) \wedge \Box_{[0,1d]}\ (\blacksquare_{=0}\ net(c) \rightarrow \Diamond_{[1min,20min]}\ net(c))\big) \rightarrow \Diamond_{[0,1d)}\ \blacklozenge_{=0}\ ssh_logout(c,\ s)$$

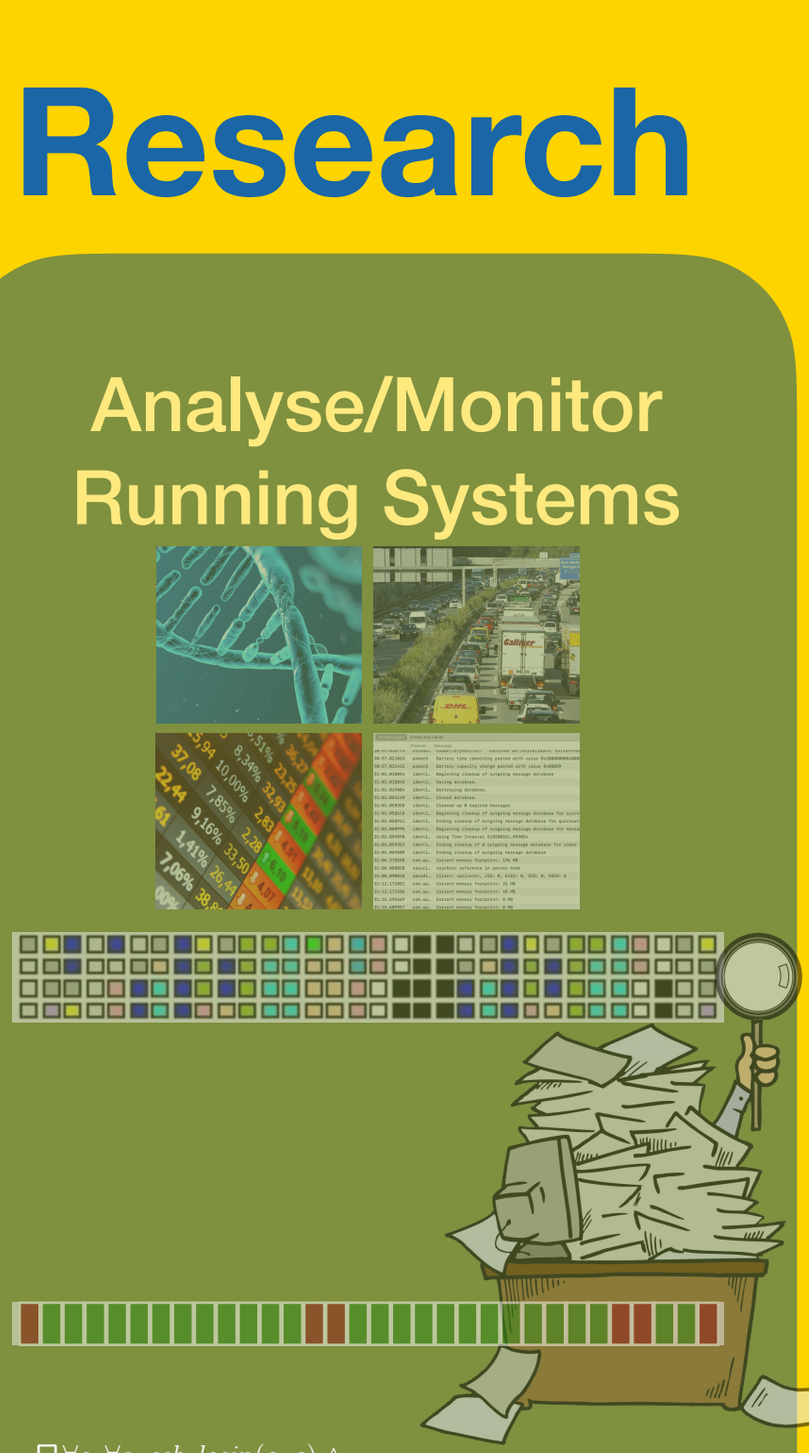
Long-running SSH sessions must not last longer than 24 hours.

My Research

Formally Verify Using Proof Assistants

X

Analyse/Monitor Running Systems



$$\Box \forall c.\ \forall s.\ ssh_login(c,\ s) \wedge \big((\Diamond_{[1min,20min]}\ net(c)) \wedge \Box_{[0,1d]}\ (\blacksquare_{=0}\ net(c) \rightarrow \Diamond_{[1min,20min]}\ net(c))\big) \rightarrow \Diamond_{[0,1d)}\ \blacklozenge_{=0}\ ssh_logout(c,\ s)$$

Long-running SSH sessions must not last longer than 24 hours.

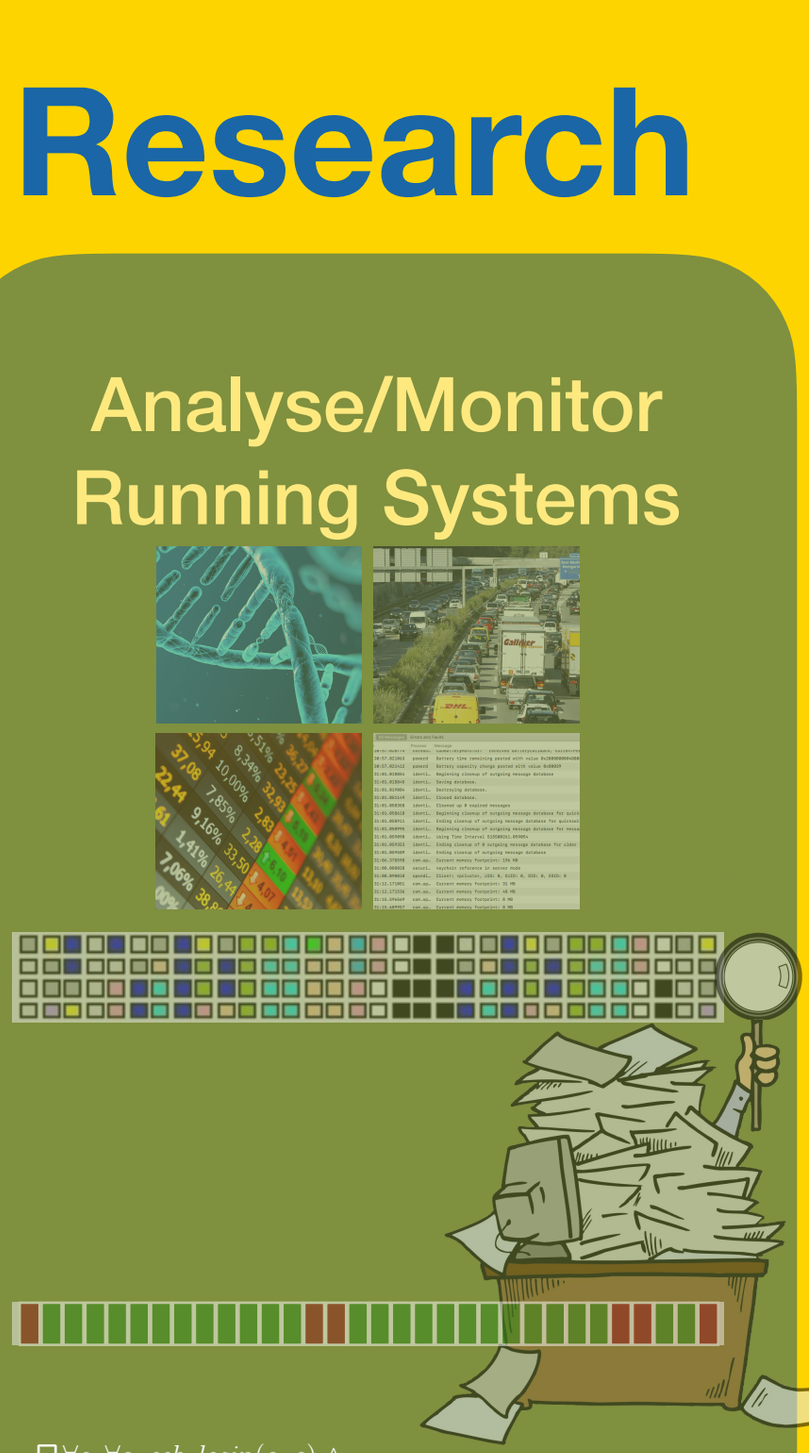
My Research

Formally Verify Using Proof Assistants

x

- Regular Expression Equivalence
- FO Resolution Prover
- Authenticated Data Structures
- Monitoring Algorithms
- Data Stream Processing
- Ordinals and Cardinals
- Gödel's Incompleteness Theorems

Analyse/Monitor Running Systems



$$\Box \forall c.\ \forall s.\ ssh_login(c,\ s) \wedge \big((\Diamond_{[1min,20min]}\ net(c)) \wedge \Box_{[0,1d]}\ (\blacksquare_{=0}\ net(c) \rightarrow \Diamond_{[1min,20min]}\ net(c))\big) \rightarrow \Diamond_{[0,1d)}\ \blacklozenge_{=0}\ ssh_logout(c,\ s)$$

Long-running SSH sessions must not last longer than 24 hours.

My Research

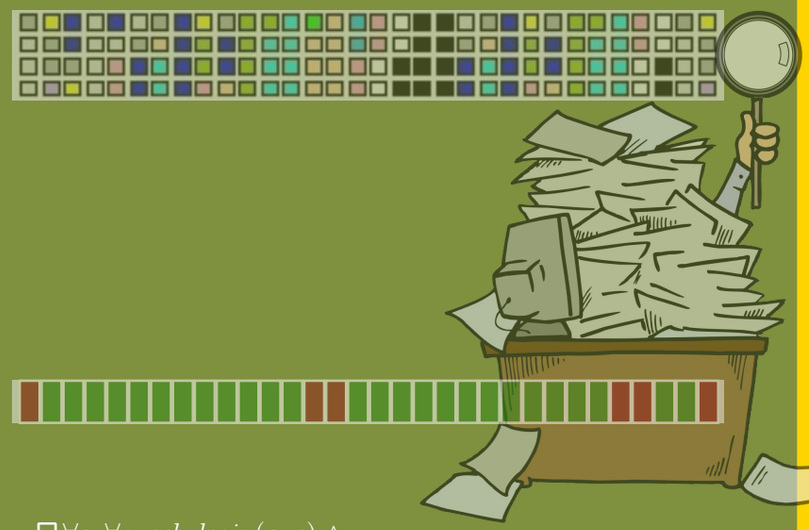
Formally Verify Using Proof Assistants

x

- Regular Expression Equivalence
- FO Resolution Prover
- Authenticated Data Structures
- Monitoring Algorithms
- Data Stream Processing
- Ordinals and Cardinals
- Gödel's Incompleteness Theorems

Improve/Extend Proof Assistants

Analyse/Monitor Running Systems



$$\Box \forall c.\ \forall s.\ ssh_login(c,\ s) \wedge ((\Diamond_{[1min,20min]}\ net(c)) \wedge \Box_{[0,1d]}\ (\blacksquare_{=0}\ net(c) \rightarrow \Diamond_{[1min,20min]}\ net(c))) \rightarrow \Diamond_{[0,1d)}\ \blacklozenge_{=0}\ ssh_logout(c,\ s)$$

Long-running SSH sessions must not last longer than 24 hours.

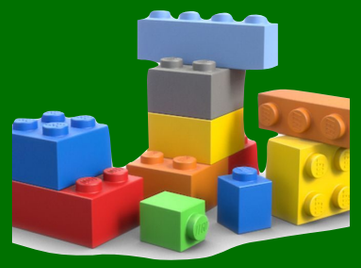
My Research

Formally Verify Using Proof Assistants

x

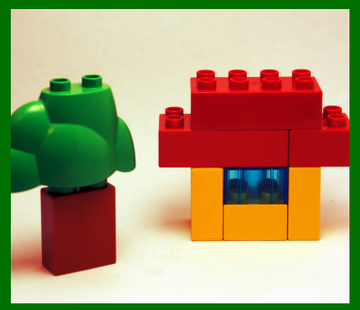
- Regular Expression Equivalence
- FO Resolution Prover
- Authenticated Data Structures
- Monitoring Algorithms
- Data Stream Processing
- Ordinals and Cardinals
- Gödel's Incompleteness Theorems

Improve/Extend Proof Assistants



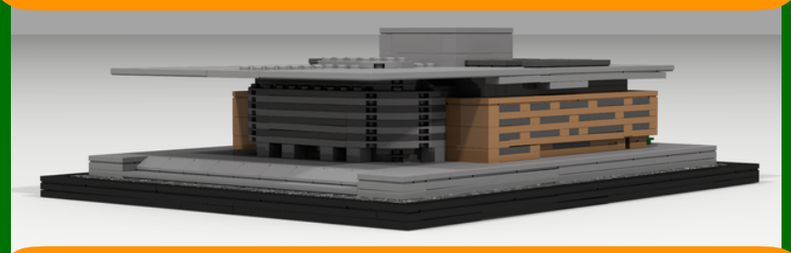
Nonrecursive subtypes

Inductive datatypes



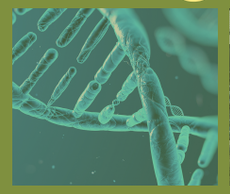
Coinductive datatypes

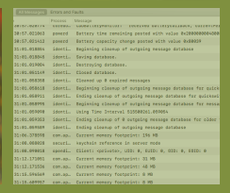
Nonuniform (co)datatypes

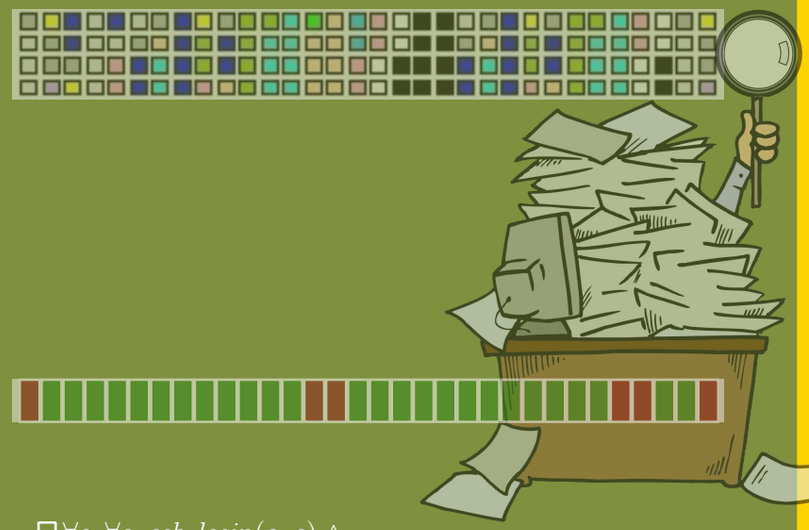


(Co)datatypes with bindings

Analyse/Monitor Running Systems







$$\Box \forall c.\ \forall s.\ ssh_login(c,\ s) \wedge \big((\Diamond_{[1min,20min]}\ net(c)) \wedge \Box_{[0,1d]}\ (\blacksquare_{=0}\ net(c) \rightarrow \Diamond_{[1min,20min]}\ net(c))\big) \rightarrow \Diamond_{[0,1d)}\ \blacklozenge_{=0}\ ssh_logout(c,\ s)$$

Long-running SSH sessions must not last longer than 24 hours.

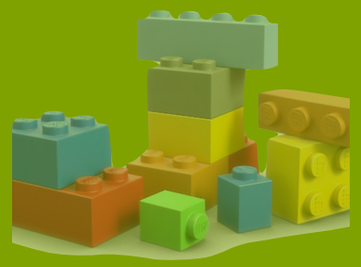
My Research

Formally Verify Using Proof Assistants

x

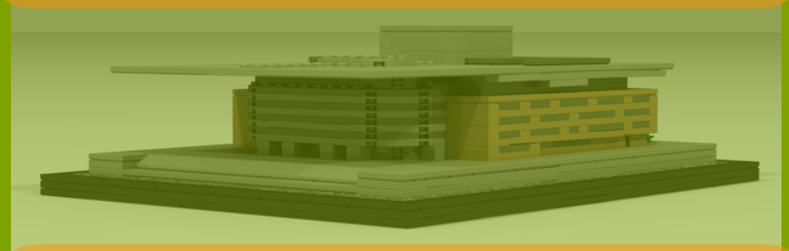
- Regular Expression Equivalence
- FO Resolution Prover
- Authenticated Data Structures
- Monitoring Algorithms
- Data Stream Processing
- Ordinals and Cardinals
- Gödel's Incompleteness Theorems

Improve/Extend Proof Assistants



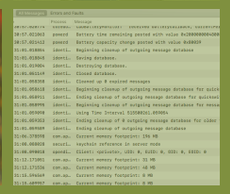
- Nonrecursive subtypes
- Inductive datatypes

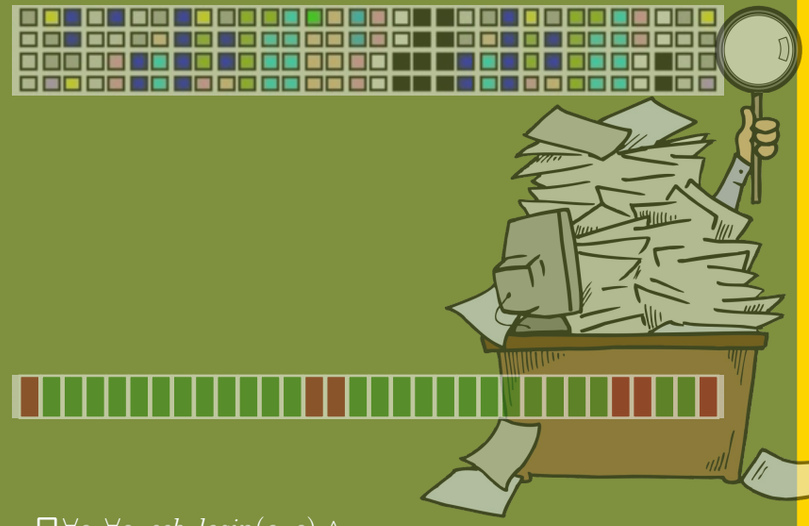
- Coinductive datatypes
- Nonuniform (co)datatypes



- (Co)datatypes with bindings

Analyse/Monitor Running Systems





$$\Box \forall c.\ \forall s.\ ssh_login(c,\ s) \wedge \big((\Diamond_{[1min,20min]}\ net(c)) \wedge \Box_{[0,1d]}\ (\blacksquare_{=0}\ net(c) \rightarrow \Diamond_{[1min,20min]}\ net(c))\big) \rightarrow \Diamond_{[0,1d)}\ \blacklozenge_{=0}\ ssh_logout(c,\ s)$$

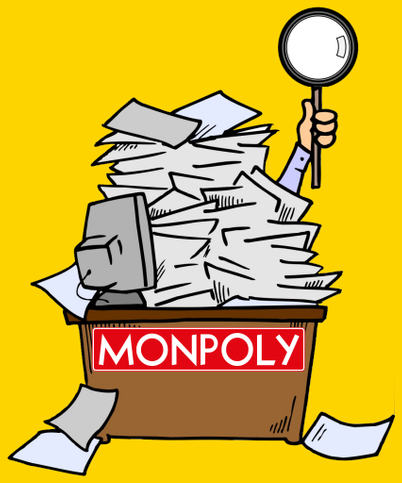
Long-running SSH sessions must not last longer than 24 hours.

```
@70  E(30,"gps") E(25,"wifi")
@100 E(20,"gps")
@170 E(20,"wifi") E(20,"bluetooth")
@230 E(30,"wifi")
@300 E(10,"wifi")
```

MONPOLY

Basin, Klaedtke, Müller, Zălinescu

J. ACM 2015




```
(e <- MAX e; (ONCE[60,120] E(e,id)))
          AND (ONCE[60,120] E(e,id))
```

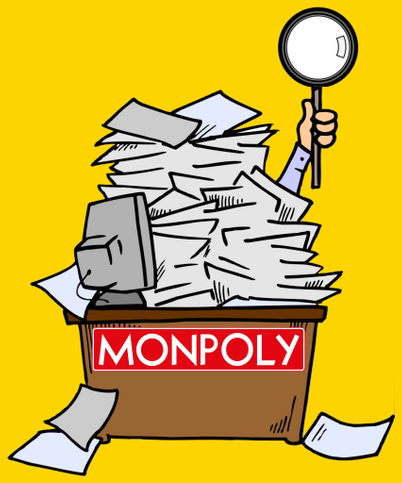
Report the module with the highest energy consumption within the second to last minute.

```
@70  E(30,"gps") E(25,"wifi")
@100 E(20,"gps")
@170 E(20,"wifi") E(20,"bluetooth")
@230 E(30,"wifi")
@300 E(10,"wifi")
```

MONPOLY

Basin, Klaedtke, Müller, Zălinescu

J. ACM 2015




```
(e <- MAX e; (ONCE[60,120] E(e,id)))
          AND (ONCE[60,120] E(e,id))
```

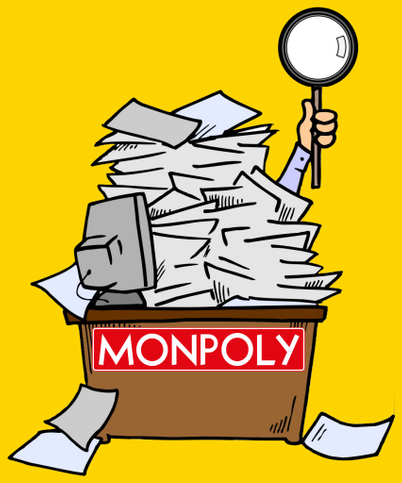
Report the module with the highest energy consumption within the second to last minute.

```
@70  E(30,"gps") E(25,"wifi")
@100 E(20,"gps")
@170 E(20,"wifi") E(20,"bluetooth")
@230 E(30,"wifi")
@300 E(10,"wifi")
```

MONPOLY

Basin, Klaedtke, Müller, Zălinescu

J. ACM 2015




```
(e <- MAX e; (ONCE[60,120] E(e,id)))
          AND (ONCE[60,120] E(e,id))
```

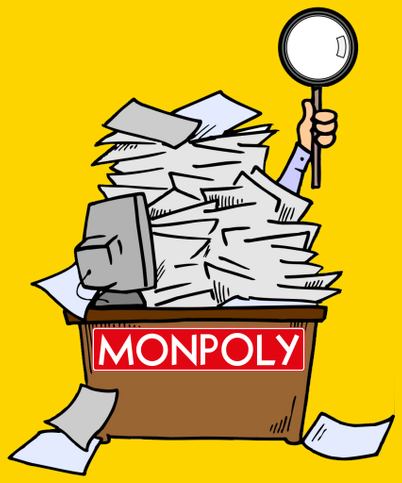
Report the module with the highest energy consumption within the second to last minute.

```
@70  E(30,"gps") E(25,"wifi")
@100 E(20,"gps")
@170 E(20,"wifi") E(20,"bluetooth")
@230 E(30,"wifi")
@300 E(10,"wifi")
```

MONPOLY

Basin, Klaedtke, Müller, Zălinescu

J. ACM 2015




```
@170 e=30 id="gps"
```

```
(e <- MAX e; (ONCE[60,120] E(e,id)))
          AND (ONCE[60,120] E(e,id))
```

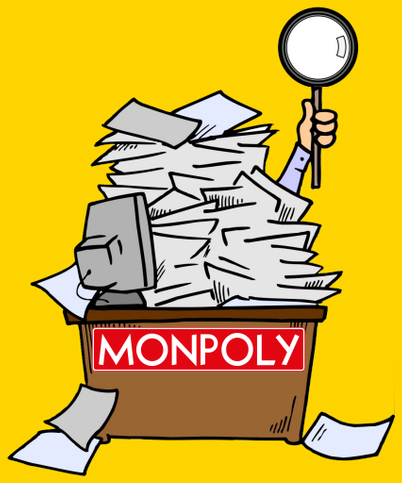
Report the module with the highest energy consumption within the second to last minute.

```
@70  E(30,"gps") E(25,"wifi")
@100 E(20,"gps")
@170 E(20,"wifi") E(20,"bluetooth")
@230 E(30,"wifi")
@300 E(10,"wifi")
```

MONPOLY

Basin, Klaedtke, Müller, Zălinescu

J. ACM 2015




```
@170 e=30 id="gps"
```

```
(e <- MAX e; (ONCE[60,120] E(e,id)))
          AND (ONCE[60,120] E(e,id))
```

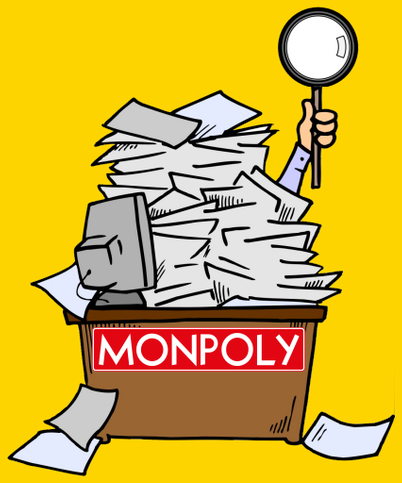
Report the module with the highest energy consumption within the second to last minute.

```
@70  E(30,"gps") E(25,"wifi")
@100 E(20,"gps")
@170 E(20,"wifi") E(20,"bluetooth")
@230 E(30,"wifi")
@300 E(10,"wifi")
```

MONPOLY

Basin, Klaedtke, Müller, Zălinescu

J. ACM 2015




```
@170 e=30 id="gps"
@300 e=30 id="wifi"
```

```
(e <- MAX e; (ONCE[60,120] E(e,id)))
          AND (ONCE[60,120] E(e,id))
```

Report the module with the highest energy consumption within the second to last minute.

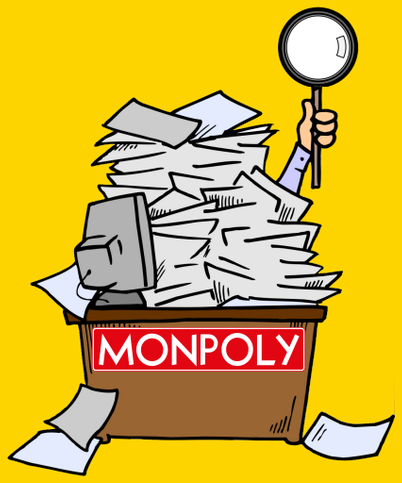
```
@70  E(30,"gps") E(25,"wifi")
@100 E(20,"gps")
@170 E(20,"wifi") E(20,"bluetooth")
@230 E(30,"wifi")
@300 E(10,"wifi")
```

MONPOLY

Basin, Klaedtke, Müller, Zălinescu

J. ACM 2015

Q: what about @230?




```
@170 e=30 id="gps"
@300 e=30 id="wifi"
```

```
(e <- MAX e; (ONCE[60,120] E(e,id)))
          AND (ONCE[60,120] E(e,id))
```

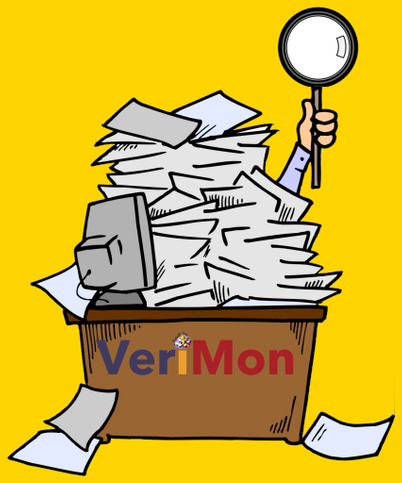
Report the module with the highest energy consumption within the second to last minute.

```
@70  E(30,"gps") E(25,"wifi")
@100 E(20,"gps")
@170 E(20,"wifi") E(20,"bluetooth")
@230 E(30,"wifi")
@300 E(10,"wifi")
```

VeriMon

Basin, Dardinier, Heimes, Krstic, Raszyk, Schneider, T., Zingg

RV 2019, IJCAR 2020, TACAS 2022




```
(e <- MAX e; (ONCE[60,120] E(e,id)))
          AND (ONCE[60,120] E(e,id))
```

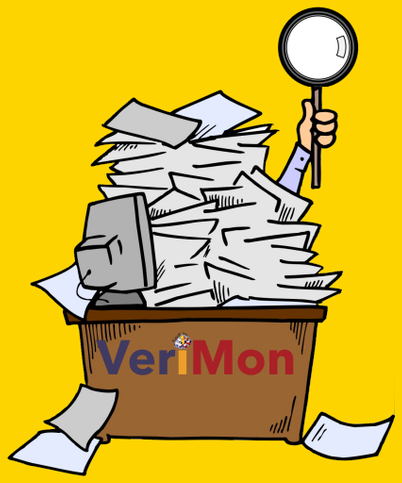
Report the module with the highest energy consumption within the second to last minute.

```
@70  E(30,"gps") E(25,"wifi")
@100 E(20,"gps")
@170 E(20,"wifi") E(20,"bluetooth")
@230 E(30,"wifi")
@300 E(10,"wifi")
```

VeriMon

Basin, Dardinier, Heimes, Krstic, Raszyk, Schneider, T., Zingg

RV 2019, IJCAR 2020, TACAS 2022




```
(e <- MAX e; (ONCE[60,120] E(e,id)))
          AND (ONCE[60,120] E(e,id))
```

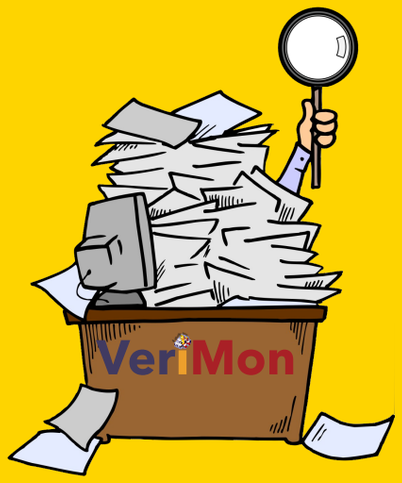
Report the module with the highest energy consumption within the second to last minute.

```
@70  E(30,"gps") E(25,"wifi")
@100 E(20,"gps")
@170 E(20,"wifi") E(20,"bluetooth")
@230 E(30,"wifi")
@300 E(10,"wifi")
```

VeriMon

Basin, Dardinier, Heimes, Krstic, Raszyk, Schneider, T., Zingg

RV 2019, IJCAR 2020, TACAS 2022




```
(e <- MAX e; (ONCE[60,120] E(e,id)))
          AND (ONCE[60,120] E(e,id))
```

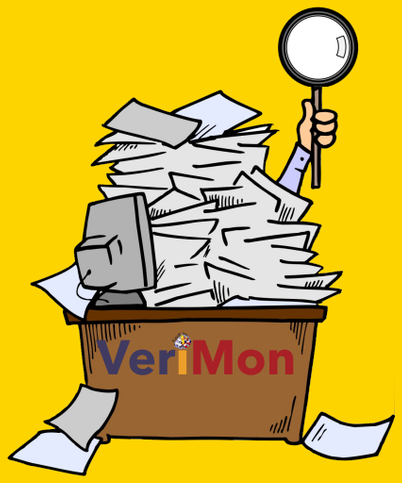
Report the module with the highest energy consumption within the second to last minute.

```
@70  E(30,"gps") E(25,"wifi")
@100 E(20,"gps")
@170 E(20,"wifi") E(20,"bluetooth")
@230 E(30,"wifi")
@300 E(10,"wifi")
```

VeriMon

Basin, Dardinier, Heimes, Krstic, Raszyk, Schneider, T., Zingg

RV 2019, IJCAR 2020, TACAS 2022




```
@170 e=30 id="gps"
```

```
(e <- MAX e; (ONCE[60,120] E(e,id)))
          AND (ONCE[60,120] E(e,id))
```

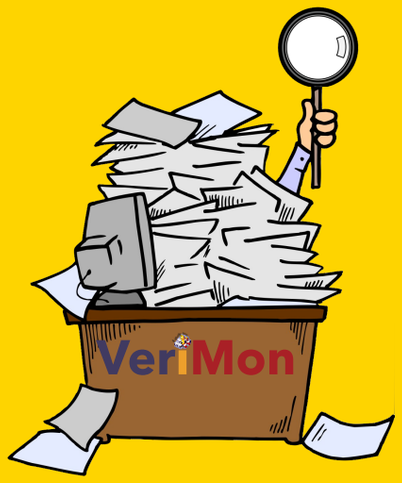
Report the module with the highest energy consumption within the second to last minute.

```
@70  E(30,"gps") E(25,"wifi")
@100 E(20,"gps")
@170 E(20,"wifi") E(20,"bluetooth")
@230 E(30,"wifi")
@300 E(10,"wifi")
```

VeriMon

Basin, Dardinier, Heimes, Krstic, Raszyk, Schneider, T., Zingg

RV 2019, IJCAR 2020, TACAS 2022




```
@170 e=30 id="gps"
@230 e=20 id="wifi"
     e=20 id="bluetooth"
```

```
(e <- MAX e; (ONCE[60,120] E(e,id)))
          AND (ONCE[60,120] E(e,id))
```

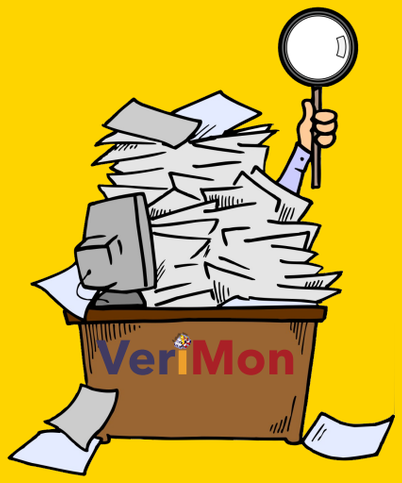
Report the module with the highest energy consumption within the second to last minute.

```
@70  E(30,"gps") E(25,"wifi")
@100 E(20,"gps")
@170 E(20,"wifi") E(20,"bluetooth")
@230 E(30,"wifi")
@300 E(10,"wifi")
```

VeriMon

Basin, Dardinier, Heimes, Krstic, Raszyk, Schneider, T., Zingg

RV 2019, IJCAR 2020, TACAS 2022




```
@170 e=30 id="gps"
@230 e=20 id="wifi"
     e=20 id="bluetooth"
@300 e=30 id="wifi"
```

```
(e <- MAX e; (ONCE[60,120] E(e,id)))
          AND (ONCE[60,120] E(e,id))
```

Report the module with the highest energy consumption within the second to last minute.

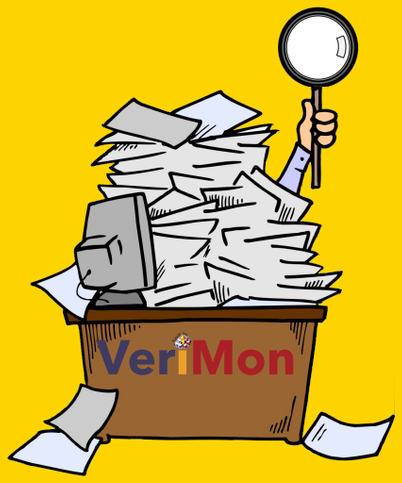
```
@70  E(30,"gps") E(25,"wifi")
@100 E(20,"gps")
@170 E(20,"wifi") E(20,"bluetooth")
@230 E(30,"wifi")
@300 E(10,"wifi")
```

VeriMon

Basin, Dardinier, Heimes, Krstic, Raszyk, Schneider, T., Zingg

RV 2019, IJCAR 2020, TACAS 2022

A: The lack of output @230 was a bug in MonPoly




```
@170 e=30 id="gps"
@230 e=20 id="wifi"
     e=20 id="bluetooth"
@300 e=30 id="wifi"
```

```
(e <- MAX e; (ONCE[60,120] E(e,id)))
          AND (ONCE[60,120] E(e,id))
```

Report the module with the highest energy consumption within the second to last minute.

Why am I so sure that

VeriMon

is right and

MONPOLY

is wrong?

Why am I so sure that

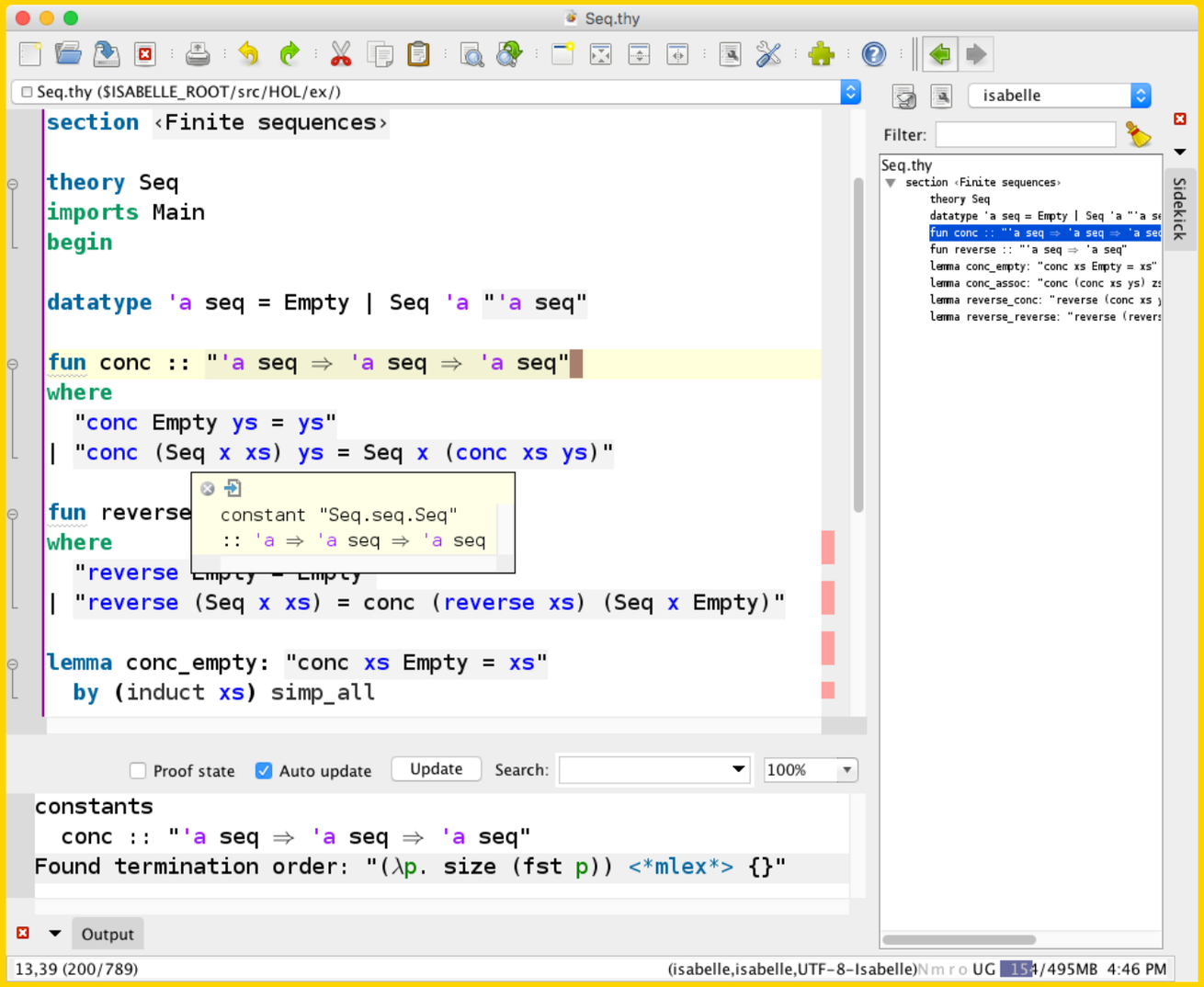
VeriMon

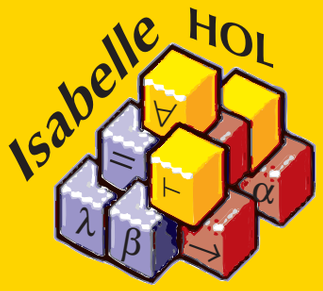
is right and

MONPOLY

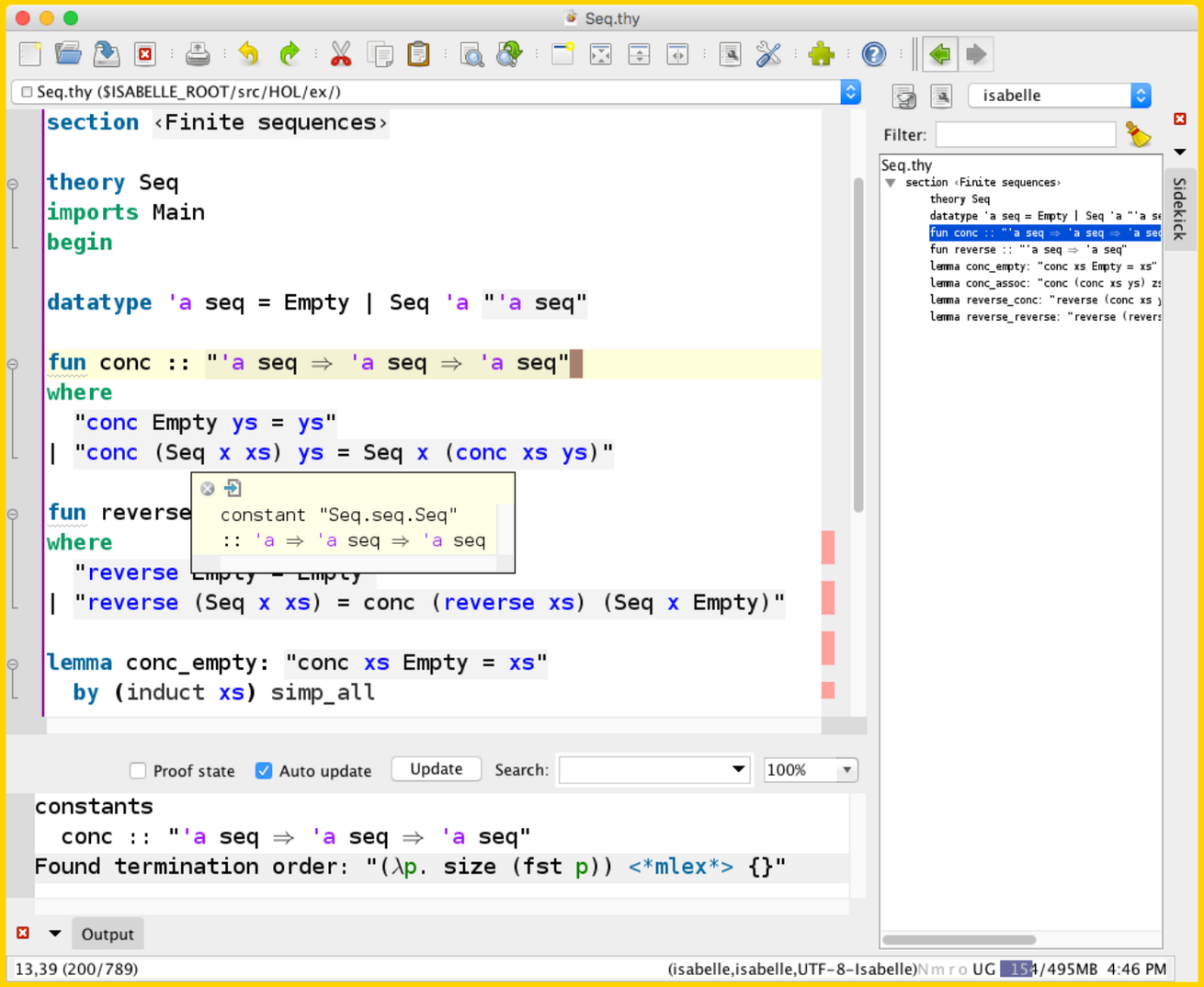
is wrong?

What Are Proof Assistants?





What Are Proof Assistants?



Coq

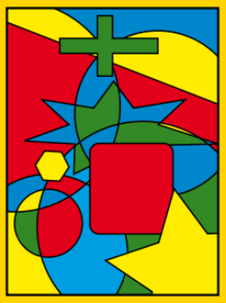
LEAN THEOREM PROVER

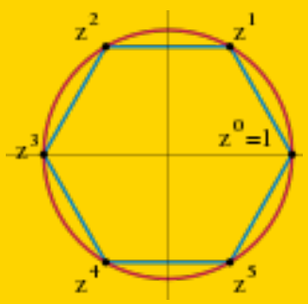
Isabelle HOL

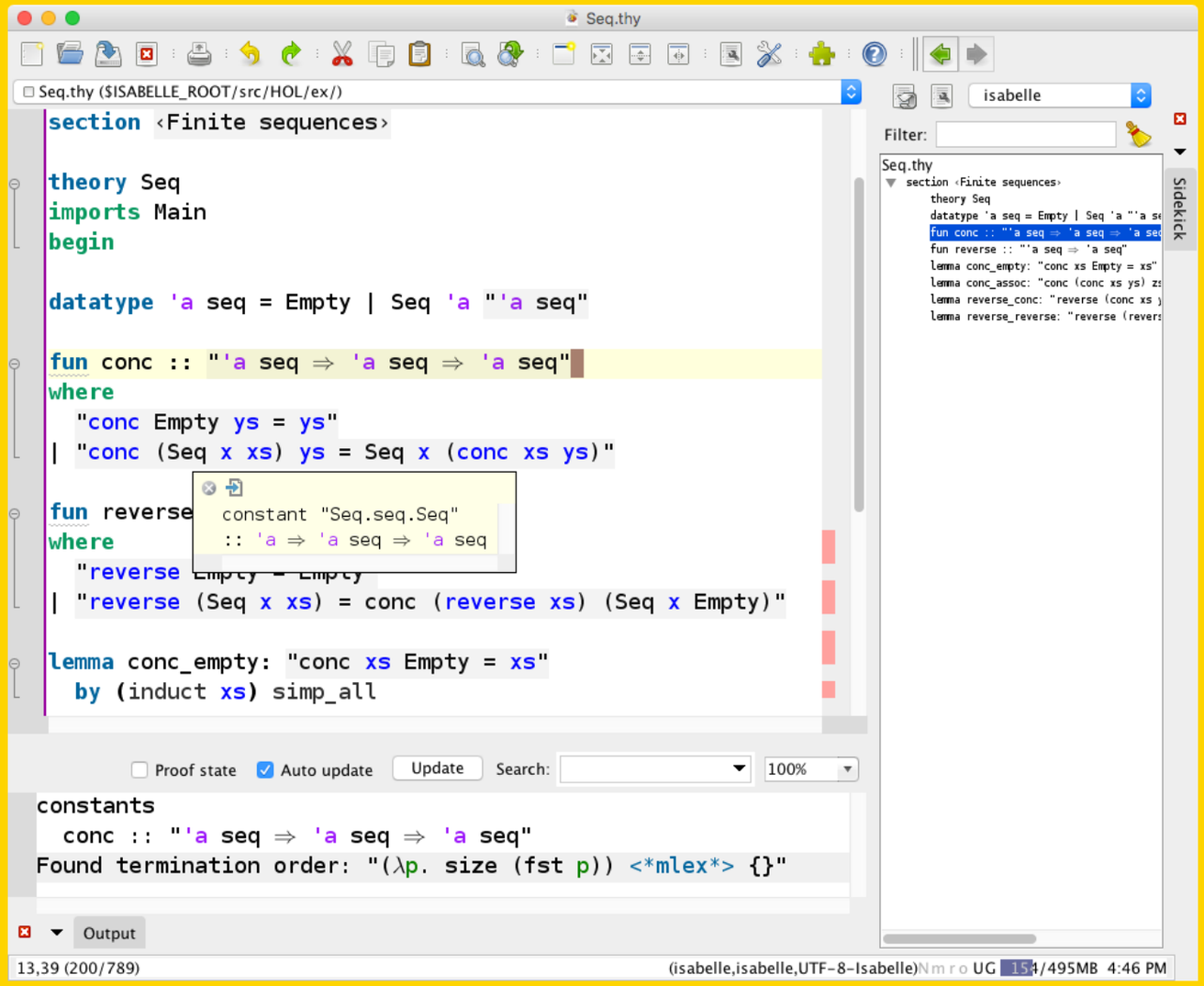
HOL4

ACL2

What Are Proof Assistants?







Coq

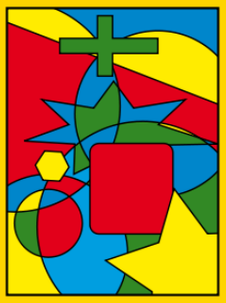
LEAN THEOREM PROVER

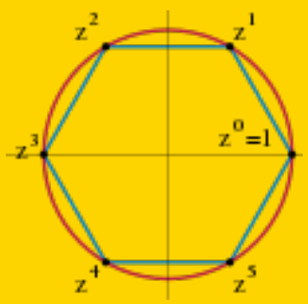
Isabelle HOL

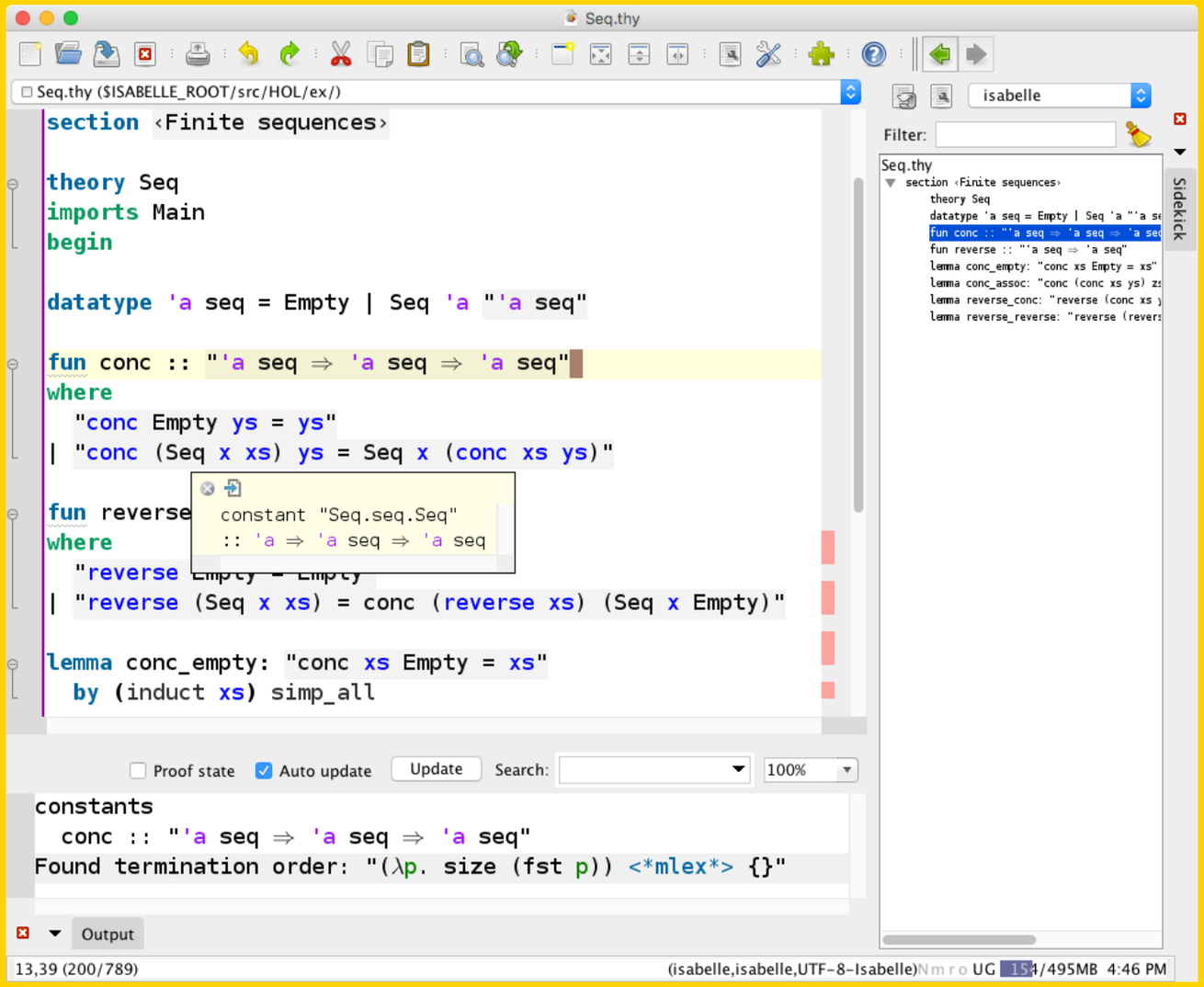
HOL4

ACL2

What Are Proof Assistants?







COMPCERT

LEAN
THEOREM PROVER

ACL2

DEMO

What Is A Monitor, Formally?

```
init :: specification → state
step :: database → state → output × state
```

What Is A Monitor, Formally?

```
φ = p(t, …, t)                          —atomic
  | NOT φ | φ OR φ | φ AND φ            —Boolean
  | ∀x. φ | ∃x. φ                       —quantifiers
  | NEXT I φ | φ UNTIL I φ              —future temporal operators
  | PREV I φ | φ SINCE I φ              —past temporal operators

I = [ℕ, ℕ ∪ {∞}]                        —intervals
t = 𝔻 | 𝕍                               —terms
```

```
init :: specification → state
step :: database → state → output × state
```

What Is A Monitor, Formally?

```
φ = p(t, …, t)                          —atomic
  | NOT φ | φ OR φ | φ AND φ            —Boolean
  | ∀x. φ | ∃x. φ                       —quantifiers
  | NEXT I φ | φ UNTIL I φ              —future temporal operators
  | PREV I φ | φ SINCE I φ              —past temporal operators

I = [ℕ, ℕ ∪ {∞}]                        —intervals
t = 𝔻 | 𝕍                               —terms
```

```
init :: specification → state
step :: database → state → output × state
```

```
@ℕ  p(𝔻, …, 𝔻)
     (𝔻, …, 𝔻)
     (𝔻, …, 𝔻)
    q(𝔻, …, 𝔻)
     (𝔻, …, 𝔻)
```

What Is A Monitor, Formally?

```
φ = p(t, …, t)                          —atomic
  | NOT φ | φ OR φ | φ AND φ            —Boolean
  | ∀x. φ | ∃x. φ                       —quantifiers
  | NEXT I φ | φ UNTIL I φ              —future temporal operators
  | PREV I φ | φ SINCE I φ              —past temporal operators

I = [ℕ, ℕ ∪ {∞}]                        —intervals
t = 𝔻 | 𝕍                               —terms
```

```
init :: specification → state
step :: database → state → output × state
```

```
@ℕ  p(𝔻, …, 𝔻)
     (𝔻, …, 𝔻)
     (𝔻, …, 𝔻)
    q(𝔻, …, 𝔻)
     (𝔻, …, 𝔻)
```

```
@ℕ  𝕍 → 𝔻
```

What Should A Monitor Compute?

$$S, V, l \models \varphi$$

What Should A Monitor Compute?

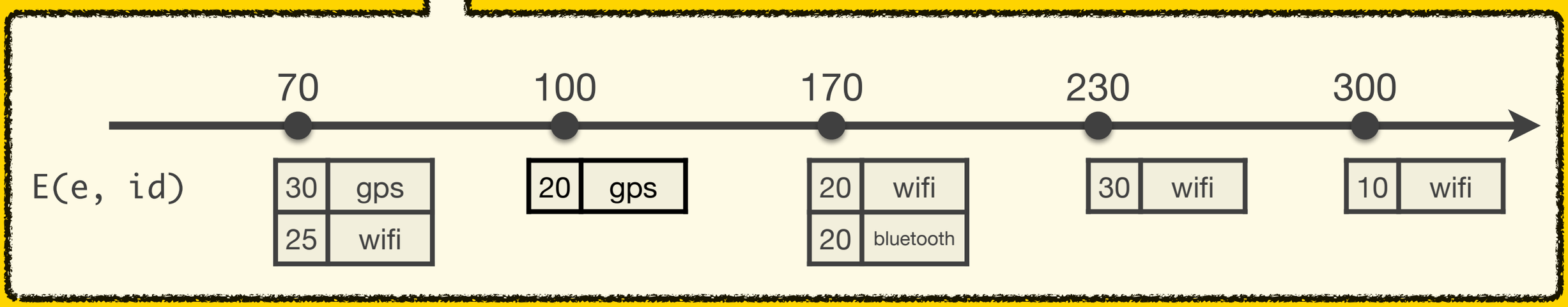
$$S, V, l \models \varphi$$

ONCE[60,120] E(e,id)

What Should A Monitor Compute?

$$S, V, I \models \varphi$$

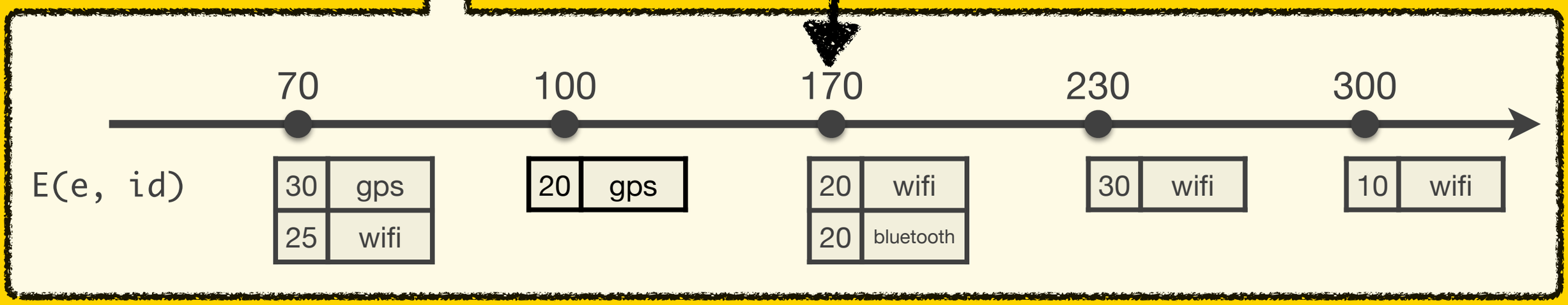
ONCE[60,120] E(e,id)



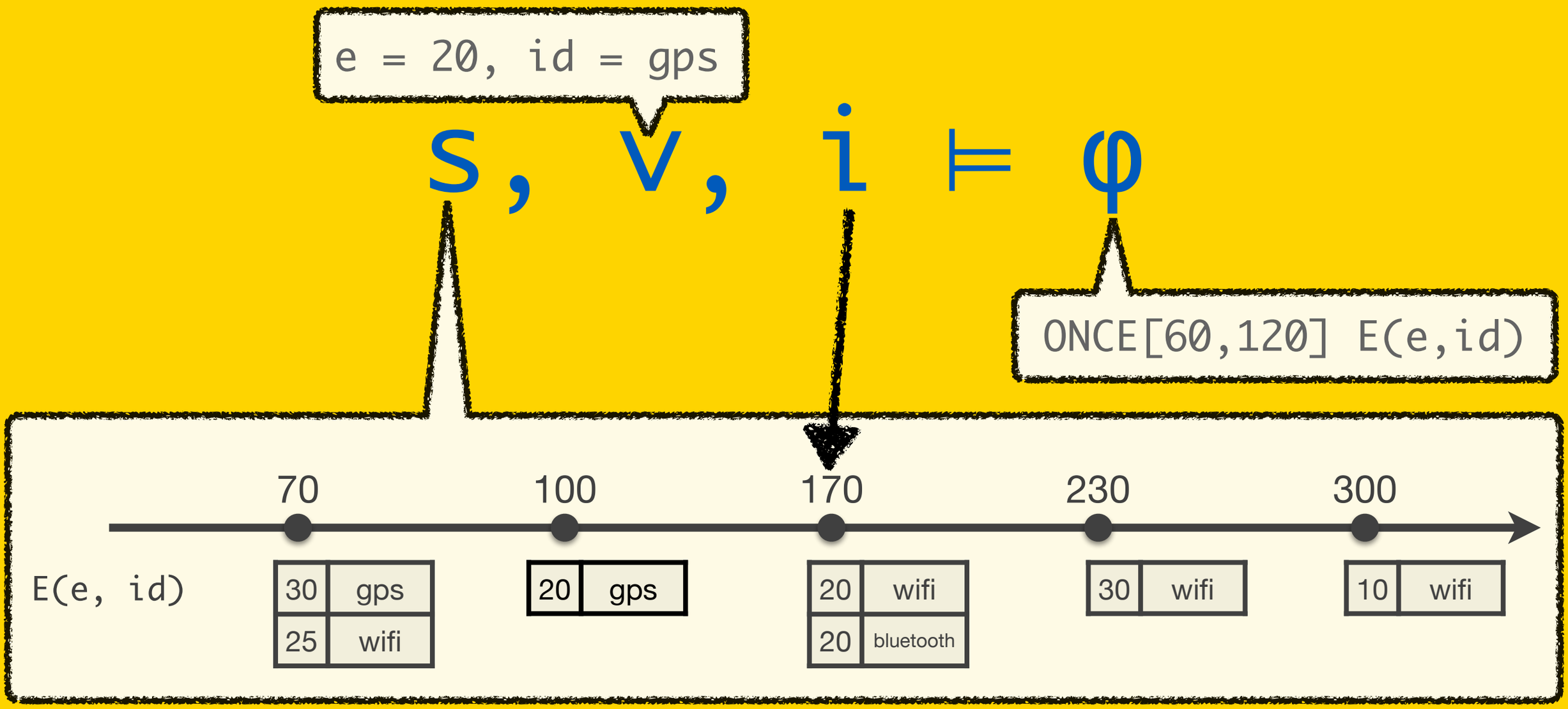
What Should A Monitor Compute?

$$S, V, l \models \varphi$$

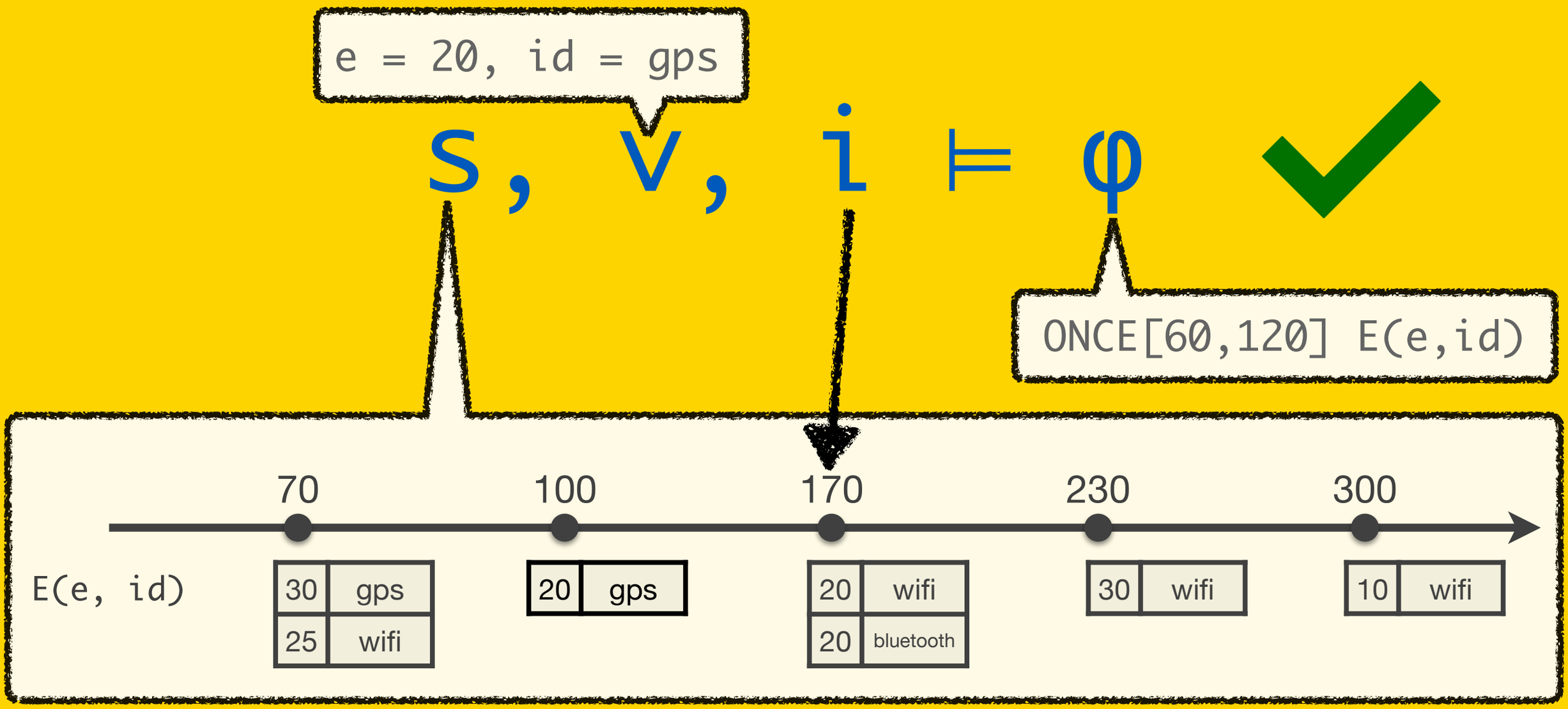
ONCE[60,120] E(e,id)



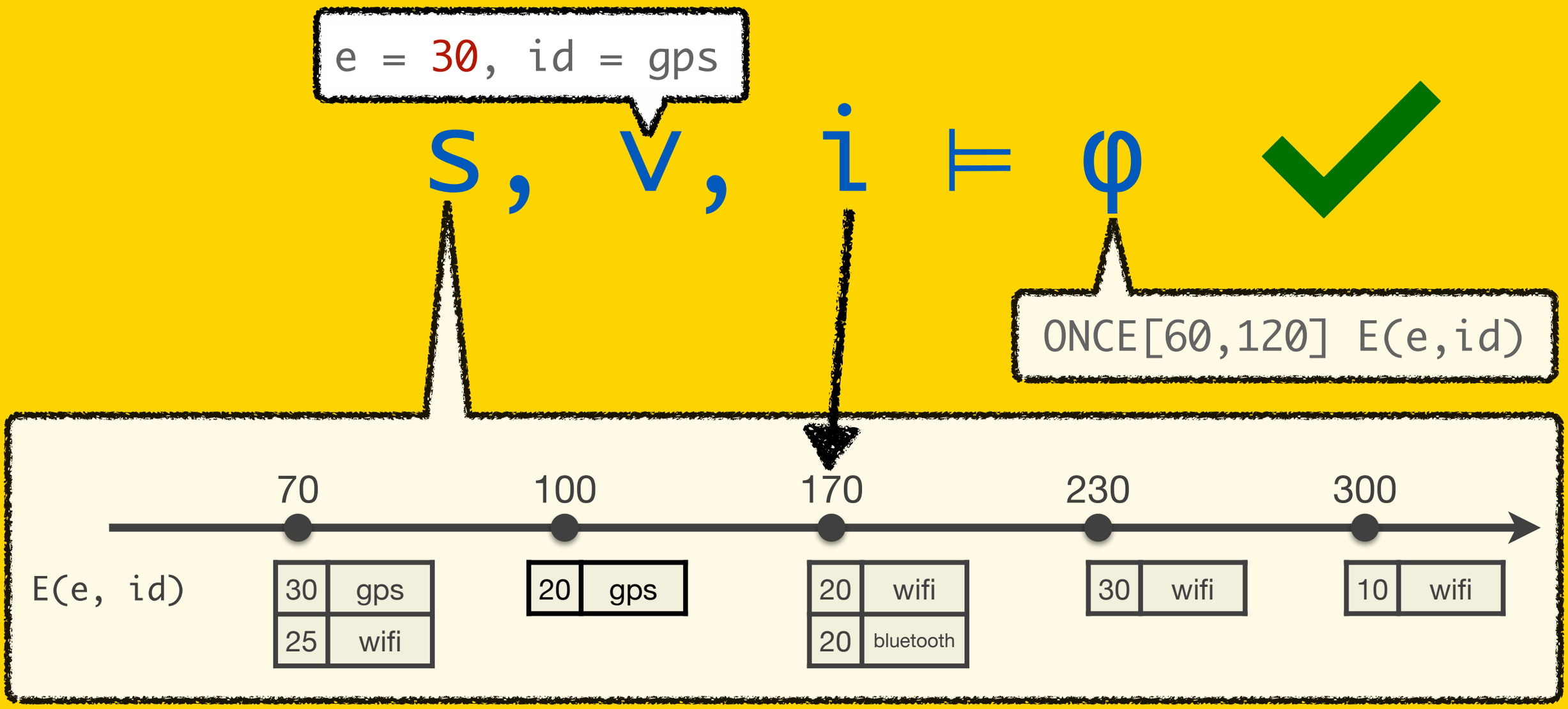
What Should A Monitor Compute?



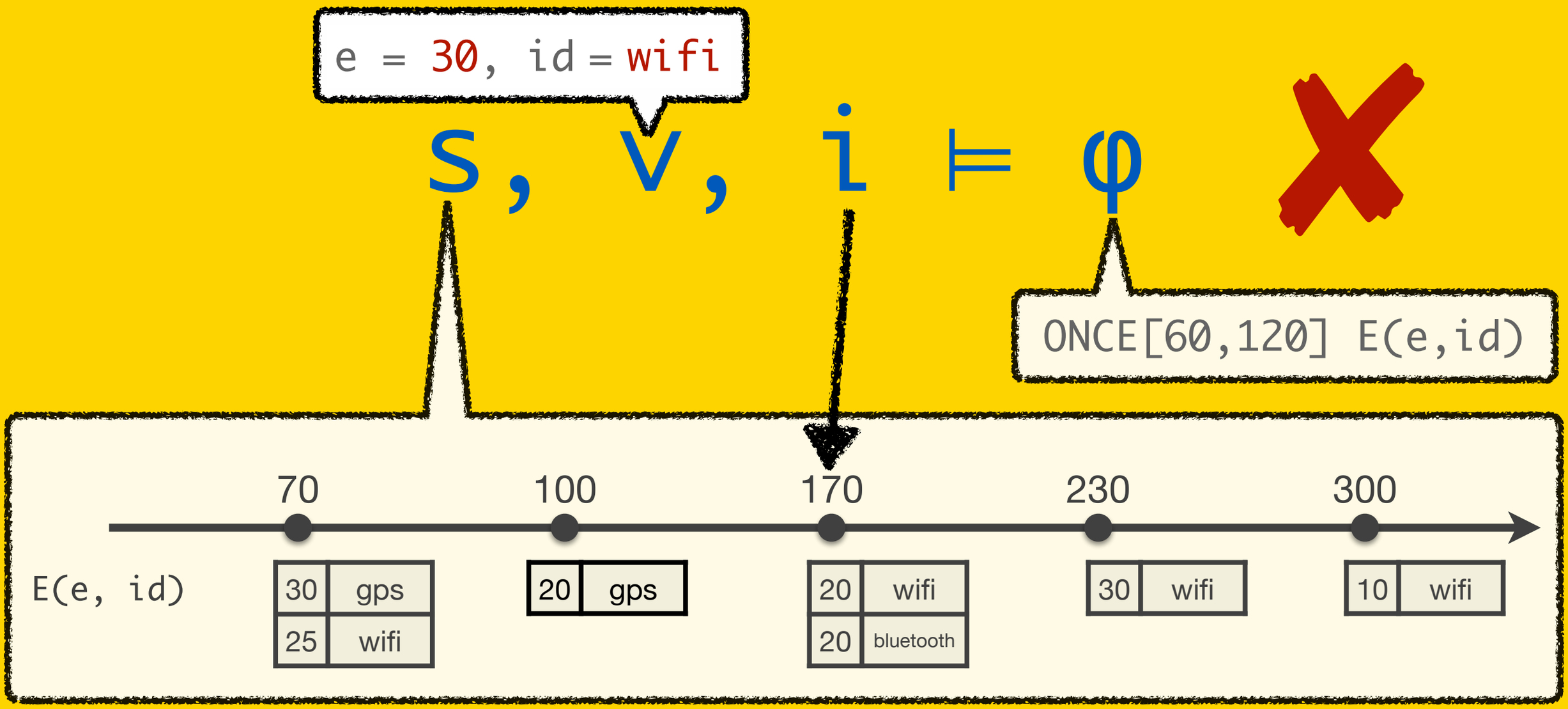
What Should A Monitor Compute?



What Should A Monitor Compute?



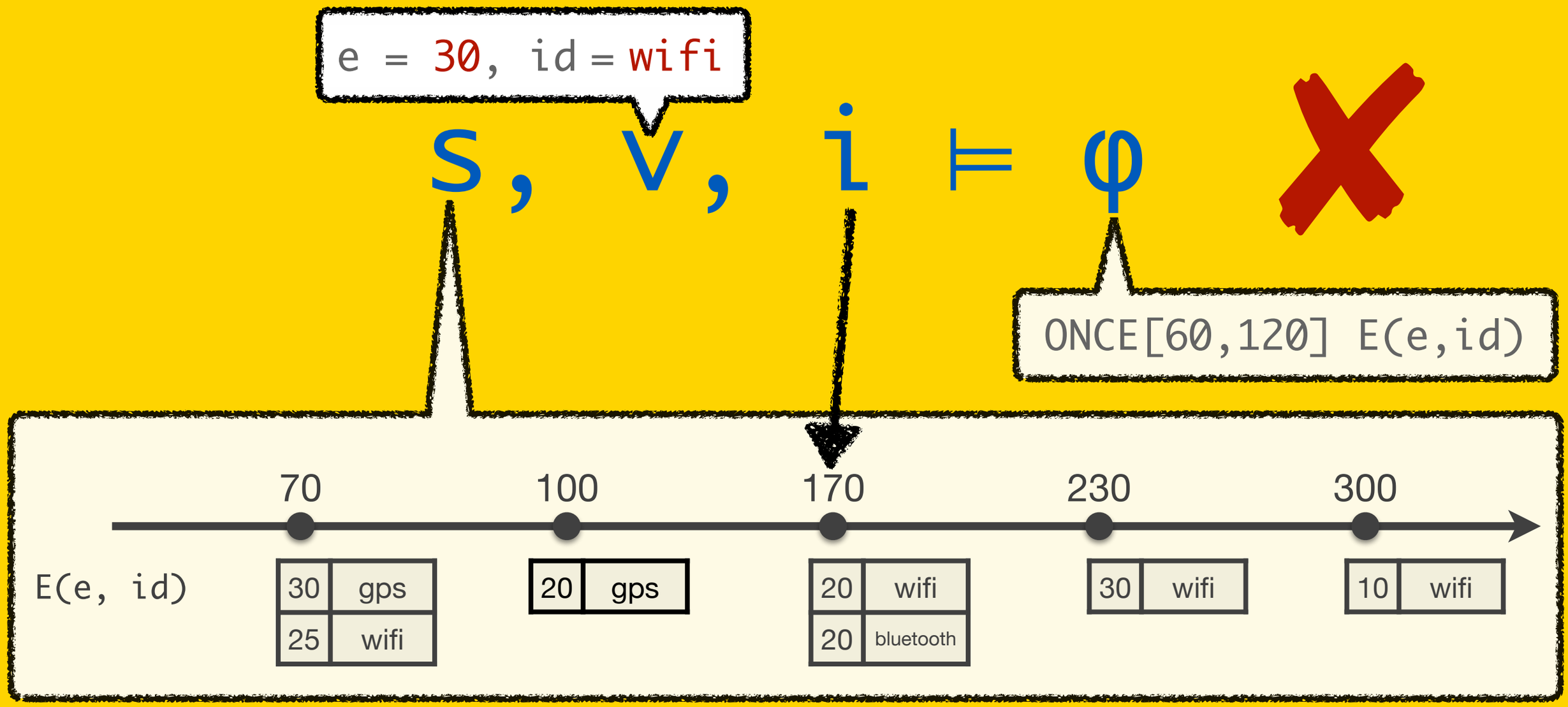
What Should A Monitor Compute?



* simplified

What Should A Monitor Compute?

`verdicts p φ` $\overset{*}{=}$ `{(i, v) | s, v, i` $\models$ `φ` for all `s` extending `p}`



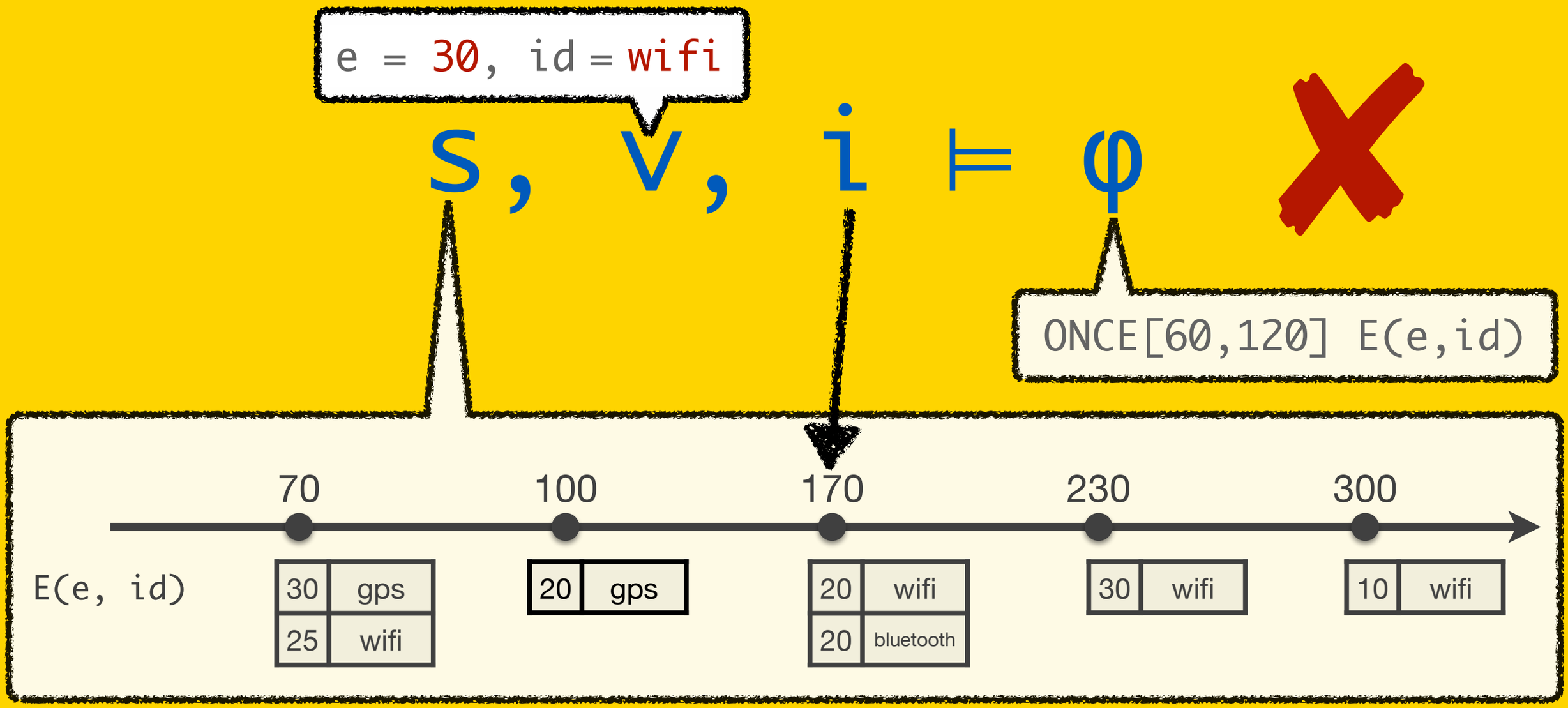
* simplified

What Should A Monitor Compute?

`verdicts p φ` $\overset{*}{=}$ {(i, v) | s, v, i ⊨ φ for all s extending p}

Theorem

1. `OK φ → INVAR φ [] (init φ)`



* simplified

What Should A Monitor Compute?

`verdicts p φ` $\overset{*}{=}$ `{(i, v) | s, v, i ⊨ φ` for all s extending p`}`

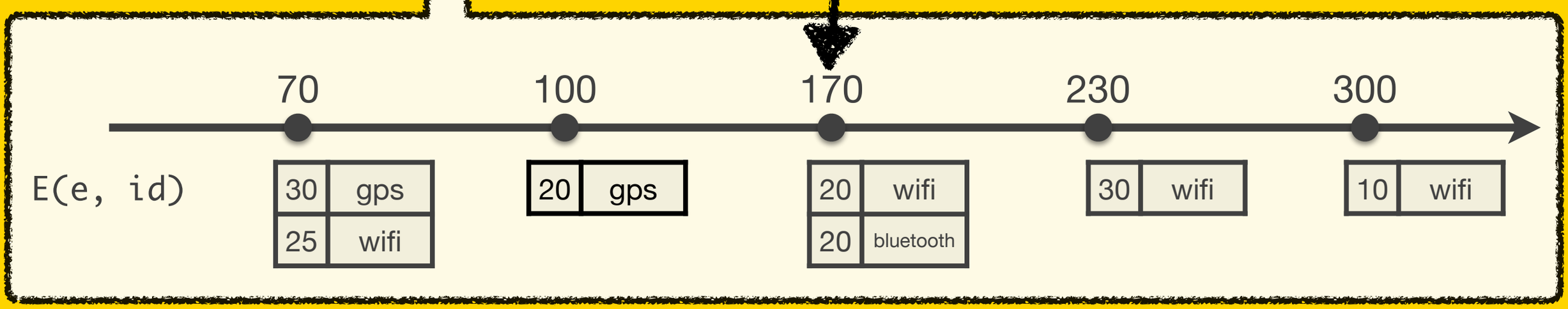
Theorem

```
1. OK φ → INVAR φ [] (init φ)
2. INVAR φ p st → step db st = (out, st') →
   INVAR φ (p·db) st'
```

`e = 30, id = wifi`

`s, v, i ⊨ φ`

`ONCE[60,120] E(e,id)`



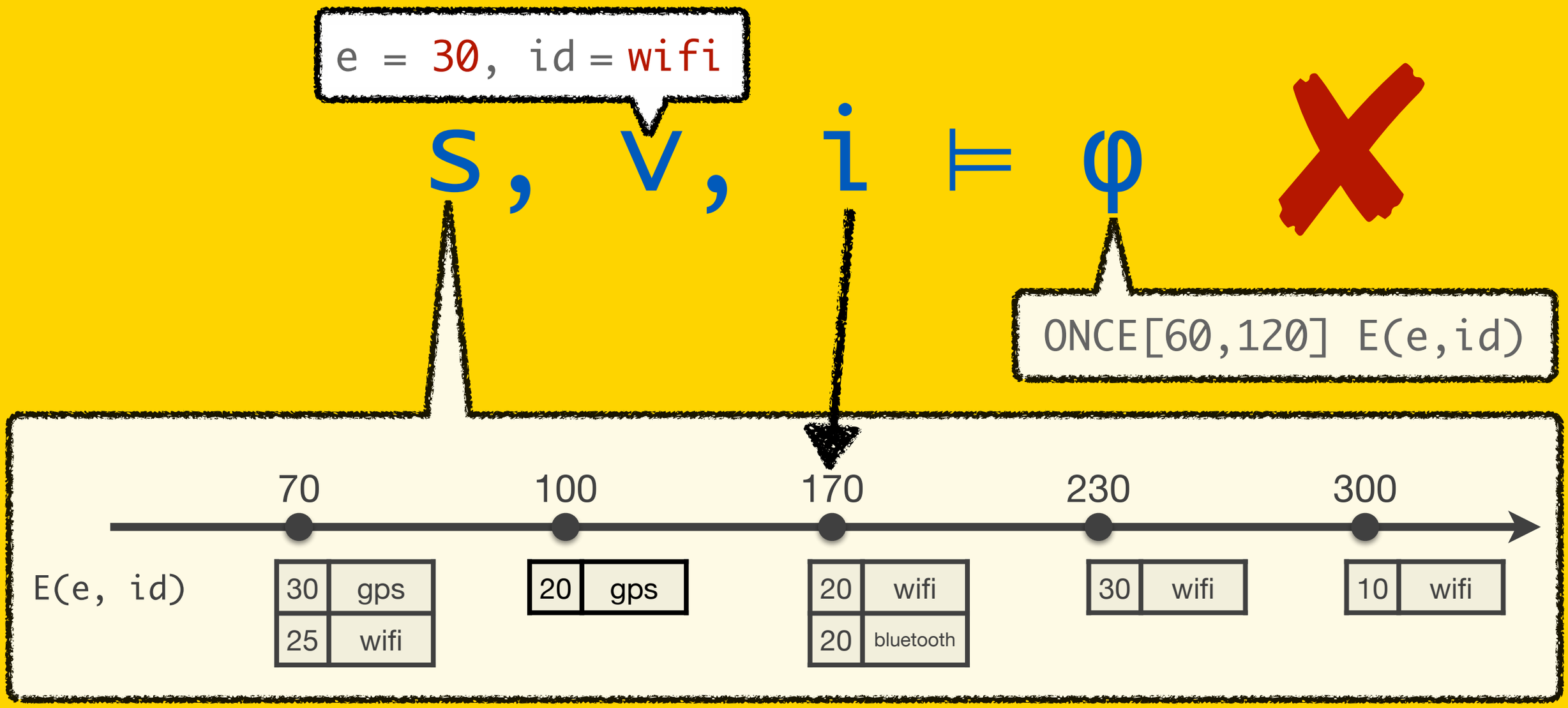
* simplified

What Should A Monitor Compute?

`verdicts p φ` $\overset{*}{=}$ `{(i, v) | s, v, i ⊨ φ` for all s extending p`}`

Theorem

```
1. OK φ → INVAR φ [] (init φ)
2. INVAR φ p st → step db st = (out, st') →
   INVAR φ (p·db) st' ∧ out = verdicts (p·db) φ \ verdicts p φ
```



The Evolution of VeriMon

Expressiveness

VeriMon

Schneider, Basin, Krstić, T.
RV 2019

```
φ = p(t, …, t)                          —atomic
  | NOT φ | φ OR φ | φ AND φ             —Boolean
  | ∀x. φ | ∃x. φ                       —quantifiers
  | NEXT I φ | φ UNTIL I φ              —future temporal operators
  | PREV I φ | φ SINCE I φ              —past temporal operators
```

```
I = [ℕ, ℕ ∪ {∞}]                        —intervals
t = 𝔻 | 𝕍                               —terms
```

The Evolution of VeriMon

Expressiveness

VeriMon

Schneider, Basin, Krstić, T.
RV 2019

```
φ = p(t, …, t) | t < t | t = t          —atomic
  | NOT φ | φ OR φ | φ AND φ            —Boolean
  | ∀x. φ | ∃x. φ                       —quantifiers
  | NEXT I φ | φ UNTIL I φ              —future temporal operators
  | PREV I φ | φ SINCE I φ              —past temporal operators
  | ωx; y1, …, yn φ                     —aggregations

ω = AVG | SUM | CNT | MIN | MAX | MED   —aggregation operators
I = [ℕ, ℕ ∪ {∞}]                        —intervals
t = 𝔻 | 𝕍 | t + t | …                   —terms
```

MONPOLY
Basin, Klaedtke, Müller, Zălinescu
J. ACM 2015

The Evolution of VeriMon

Expressiveness

VeriMon

Schneider, Basin, Krstić, T.
RV 2019

VeriMon+

Basin, Dardinier, Heimes, Krstić, Raszyk, Schneider, T.
IJCAR 2020

```
φ = p(t, …, t) | t < t | t = t                  —atomic
  | NOT φ | φ OR φ | φ AND φ                    —Boolean
  | ∀x. φ | ∃x. φ                               —quantifiers
  | NEXT I φ | φ UNTIL I φ                      —future temporal operators
  | PREV I φ | φ SINCE I φ                      —past temporal operators
  | ωx; y1, …, yn φ                             —aggregations
```

```
ω = AVG | SUM | CNT | MIN | MAX | MED            —aggregation operators
I = [ℕ, ℕ ∪ {∞}]                                —intervals
t = 𝔻 | 𝕍 | t + t | …                           —terms
```

MONPOLY

Basin, Klaedtke, Müller, Zălinescu
J. ACM 2015

The Evolution of VeriMon

Expressiveness

VeriMon

Schneider, Basin, Krstić, T.
RV 2019

VeriMon+

Basin, Dardinier, Heimes, Krstić, Raszyk, Schneider, T.
IJCAR 2020

```
φ = p(t, …, t) | t < t | t = t                    —atomic
  | NOT φ | φ OR φ | φ AND φ                      —Boolean
  | ∀x. φ | ∃x. φ                                 —quantifiers
  | NEXT I φ | φ UNTIL I φ | MATCHF I r           —future temporal operators
  | PREV I φ | φ SINCE I φ | MATCHP I r           —past temporal operators
  | ωx; y1, …, yn φ                               —aggregations

r = φ? | r + r | rr | r*                          —regular expressions
ω = AVG | SUM | CNT | MIN | MAX | MED              —aggregation operators
I = [ℕ, ℕ ∪ {∞}]                                  —intervals
t = 𝔻 | 𝕍 | t + t | …                             —terms
```

MONPOLY

Basin, Klaedtke, Müller, Zălinescu
J. ACM 2015

The Evolution of VeriMon

Expressiveness

VeriMon

Schneider, Basin, Krstić, T.
RV 2019

VeriMon^{+}

Basin, Dardinier, Heimes, Krstić, Raszyk, Schneider, T.
IJCAR 2020

VeriMon^{++}

Zingg, Krstić, Raszyk, Schneider, T.
TACAS 2022

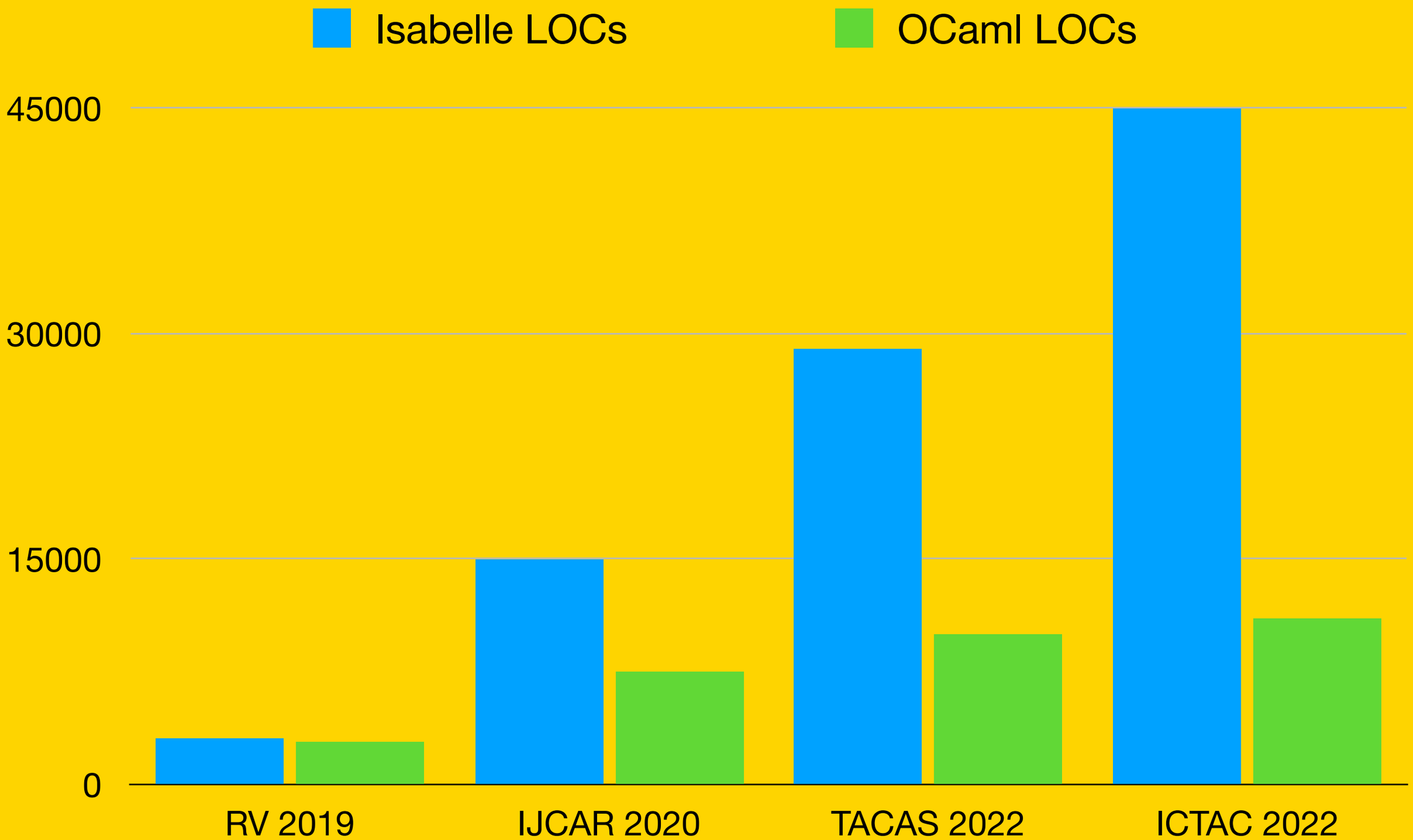
```
φ = p(t, …, t) | t < t | t = t                    —atomic
  | NOT φ | φ OR φ | φ AND φ                      —Boolean
  | ∀x. φ | ∃x. φ                                 —quantifiers
  | NEXT I φ | φ UNTIL I φ | MATCHF I r           —future temporal operators
  | PREV I φ | φ SINCE I φ | MATCHP I r           —past temporal operators
  | ωx; y1, …, yn φ                               —aggregations
  | LET p(𝕍, …, 𝕍) = φ IN φ                       —(non-recursive) let operator
  | LETPAST p(𝕍, …, 𝕍) = φ IN φ                   —past-recursive let operator
r = φ? | r + r | rr | r*                          —regular expressions
ω = AVG | SUM | CNT | MIN | MAX | MED             —aggregation operators
I = [ℕ, ℕ ∪ {∞}]                                  —intervals
t = 𝔻 | 𝕍 | t + t | …                             —terms
```

MONPOLY

Basin, Klaedtke, Müller, Zălinescu
J. ACM 2015

The Evolution of VeriMon

Statistics



The Evolution of VeriMon

Efficiency

VeriMon

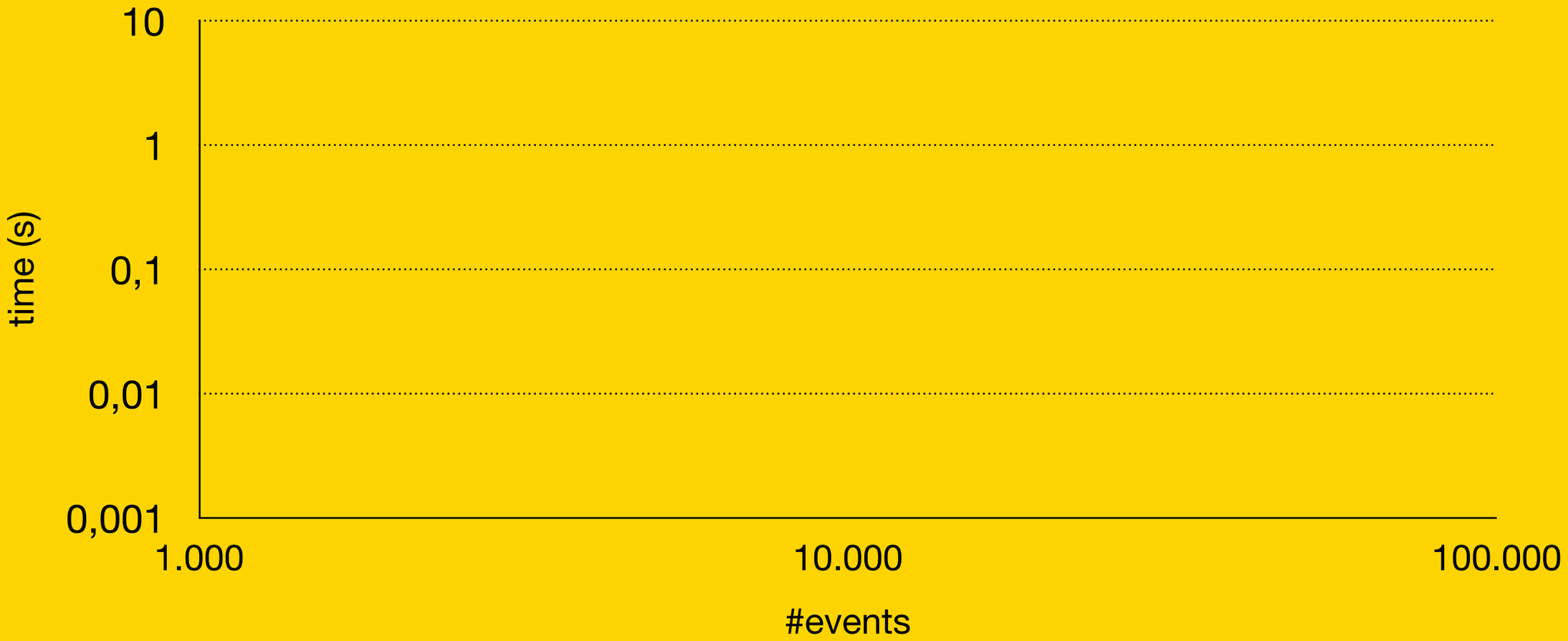
Schneider, Basin, Krstić, T.
RV 2019

MONPOLY

Basin, Klaedtke, Müller, Zălinescu
J. ACM 2015

VeriMon+

Basin, Dardinier, Heimes, Krstić, Raszyk, Schneider, T.
IJCAR 2020



The Evolution of VeriMon

Efficiency

VeriMon

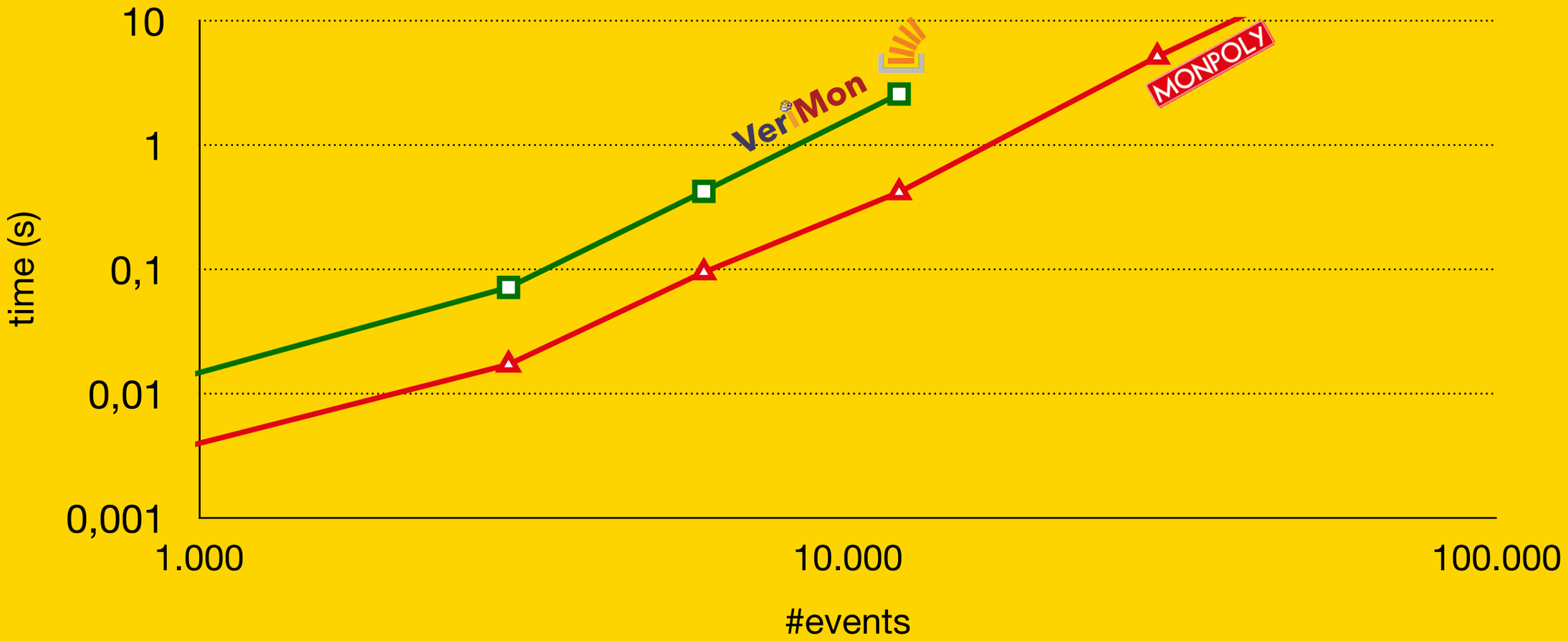
Schneider, Basin, Krstić, T.
RV 2019

MONPOLY

Basin, Klaedtke, Müller, Zălinescu
J. ACM 2015

VeriMon+

Basin, Dardinier, Heimes, Krstić, Raszyk, Schneider, T.
IJCAR 2020



The Evolution of VeriMon

Efficiency

VeriMon

Schneider, Basin, Krstić, T.
RV 2019

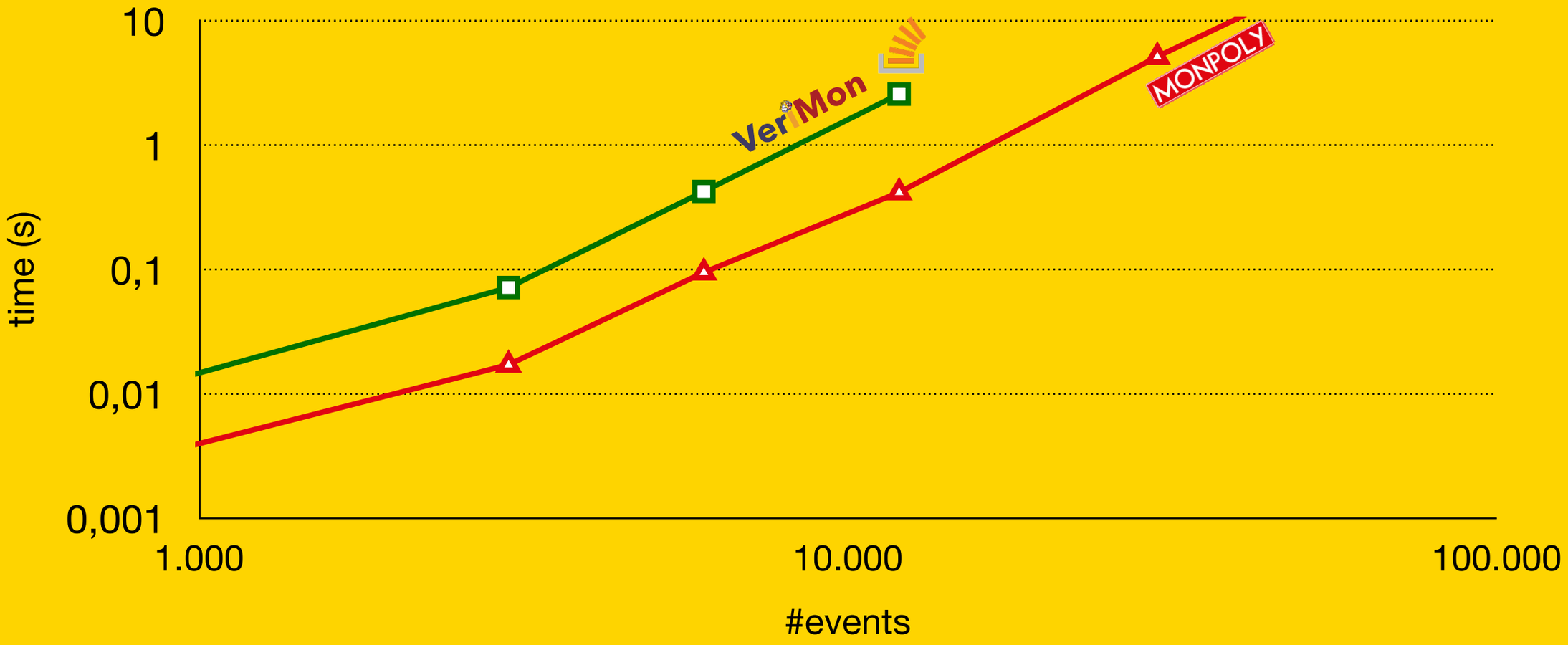
MONPOLY

Basin, Klaedtke, Müller, Zălinescu
J. ACM 2015

VeriMon+

Basin, Dardinier, Heimes, Krstić, Raszyk, Schneider, T.

B.Sc. thesis 12 ECTS
Extending and Optimizing a Verified Monitor for MFOTL



The Evolution of VeriMon

Efficiency

VeriMon

Schneider, Basin, Krstić, T.
RV 2019

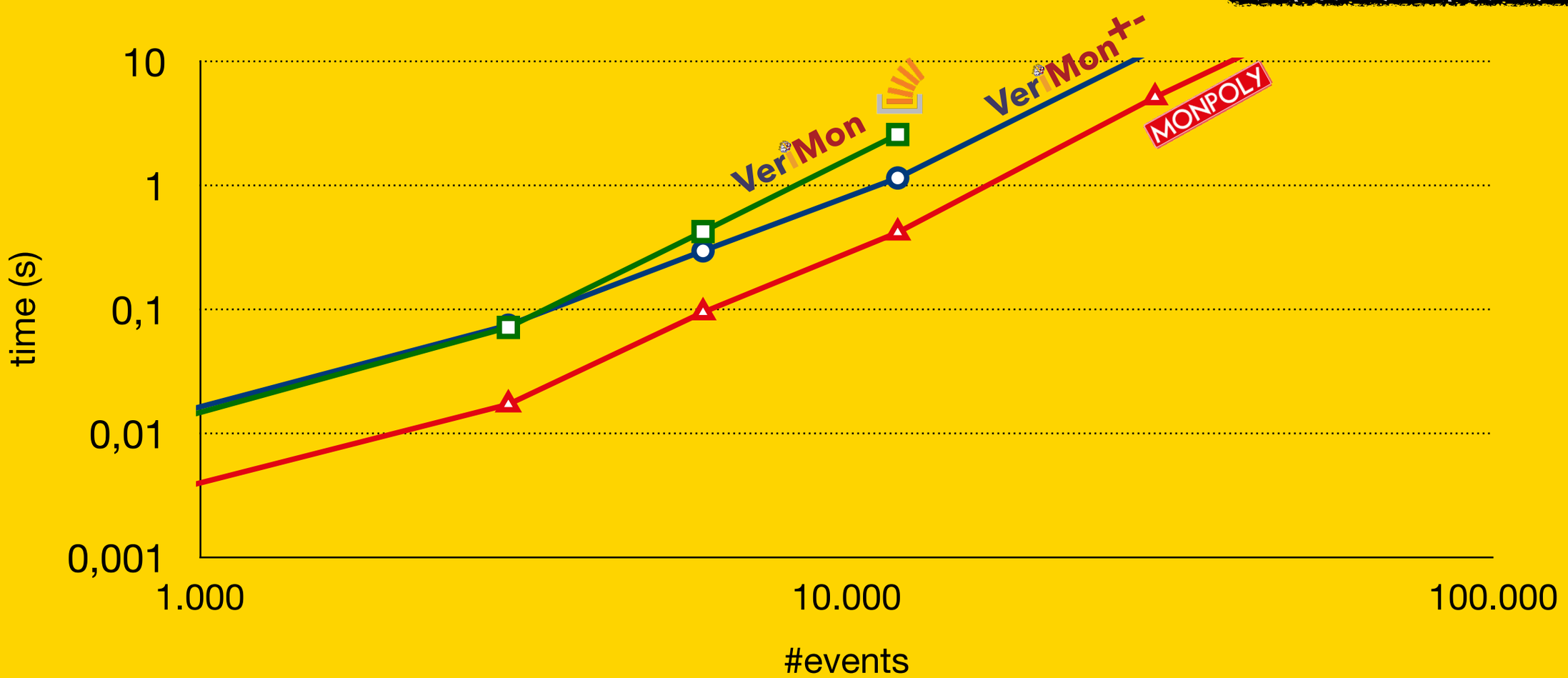
MONPOLY

Basin, Klaedtke, Müller, Zălinescu
J. ACM 2015

VeriMon+

Basin, Dardinier, Heimes, Krstić, Raszyk, Schneider, T.

B.Sc. thesis 12 ECTS
Extending and Optimizing a Verified Monitor for MFOTL



The Evolution of VeriMon

Efficien

M.Sc. project 15 ECTS
Formalizing Multiway-Join Algorithms in Isabelle/HOL

VeriMon

Schneider, Basin, Krstić, T.
RV 2019

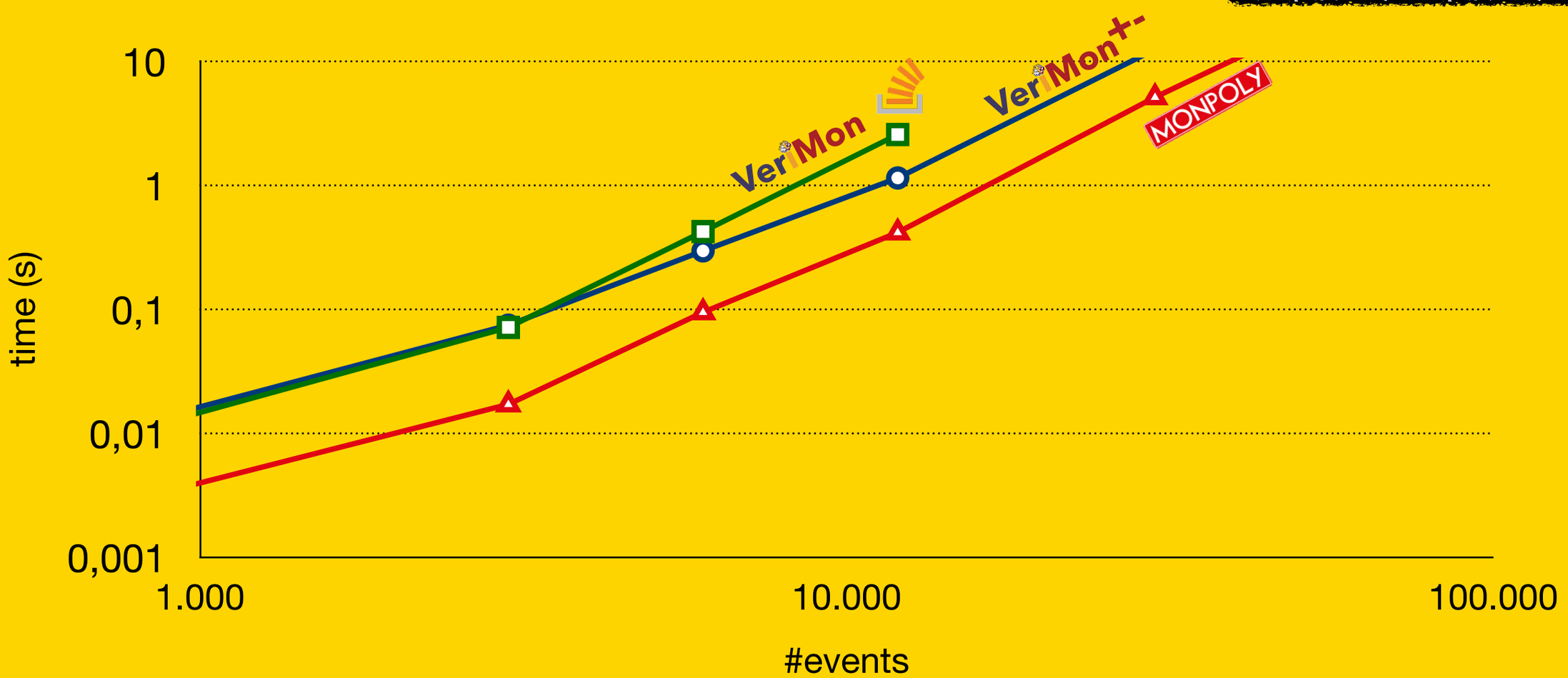
MONPO

Basin, Klaedtke, Müller, Zălinescu
J. ACM 2015

Mon+

Basin, Dardinier, Heimes, Krstić, Raszyk, Schneider, T.

B.Sc. thesis 12 ECTS
Extending and Optimizing a Verified Monitor for MFOTL



The Evolution of VeriMon

Efficien

M.Sc. project 15 ECTS
Formalizing Multiway-Join Algorithms in Isabelle/HOL

VeriMon

Schneider, Basin, Krstić, T.
RV 2019

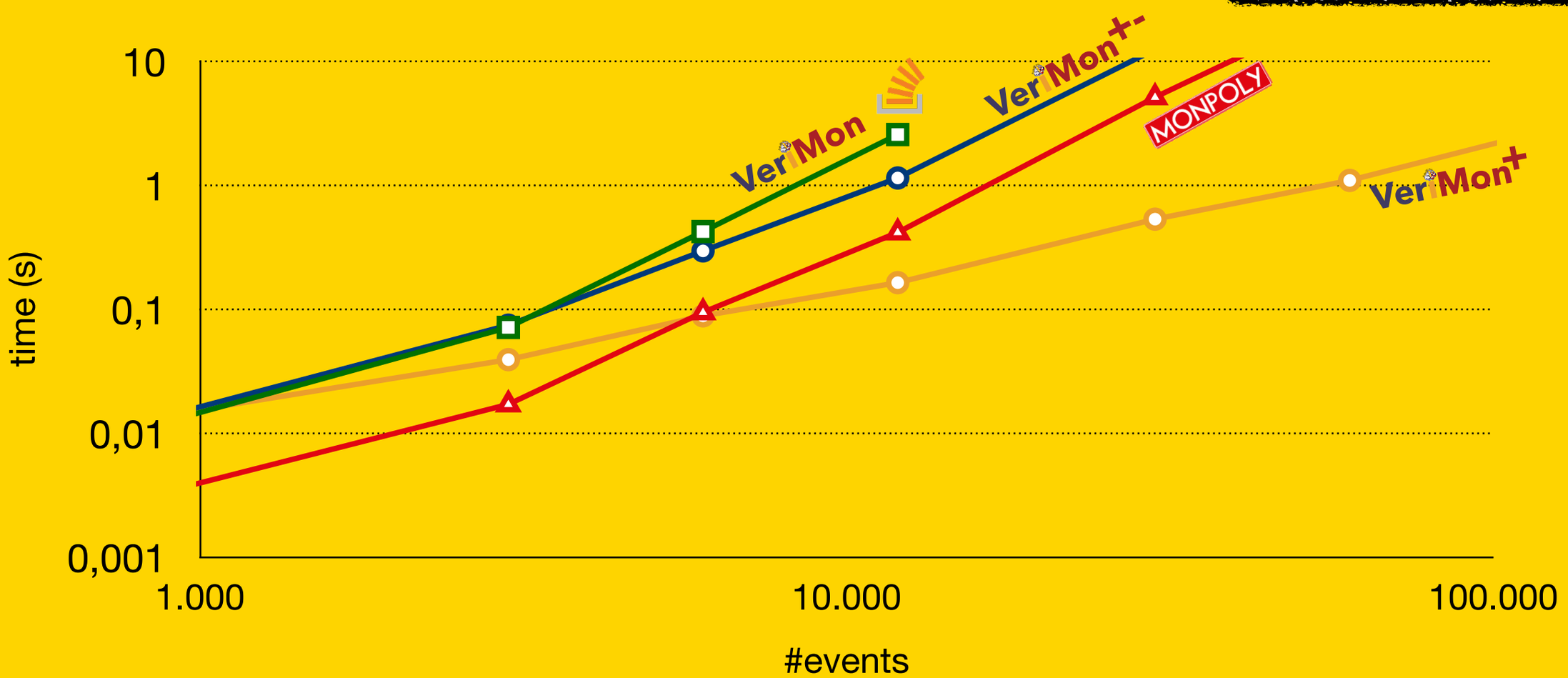
MONPO

Basin, Klaedtke, Müller, Zălinescu
J. ACM 2015

Mon+

Basin, Dardinier, Heimes, Krstić, Raszyk, Schneider, T.

B.Sc. thesis 12 ECTS
Extending and Optimizing a Verified Monitor for MFOTL

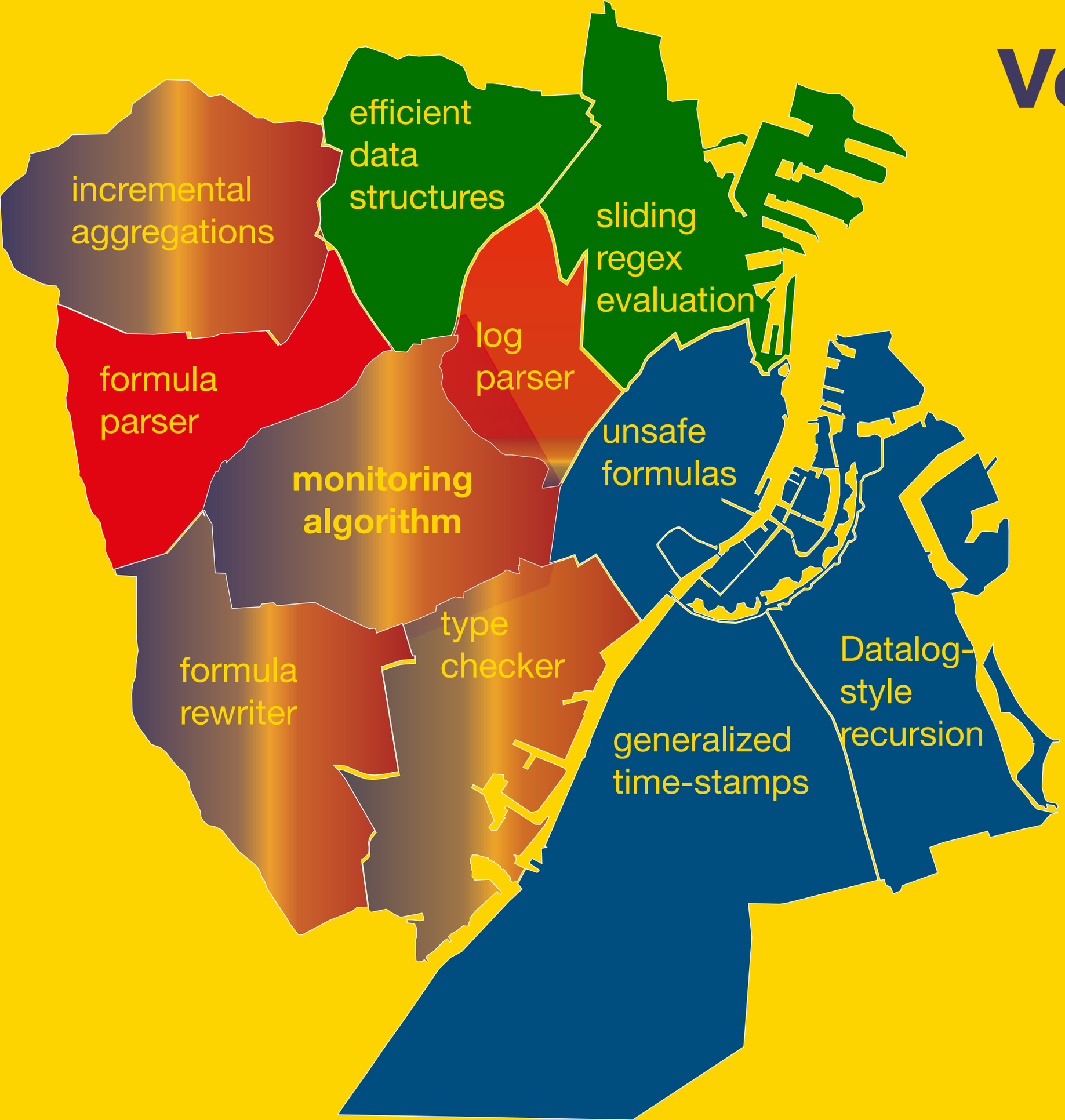


DEMO

VeriMon

https://traytel.bitbucket.io/verimon

VeriMon+++



done

trustworthiness

efficiency

expressiveness

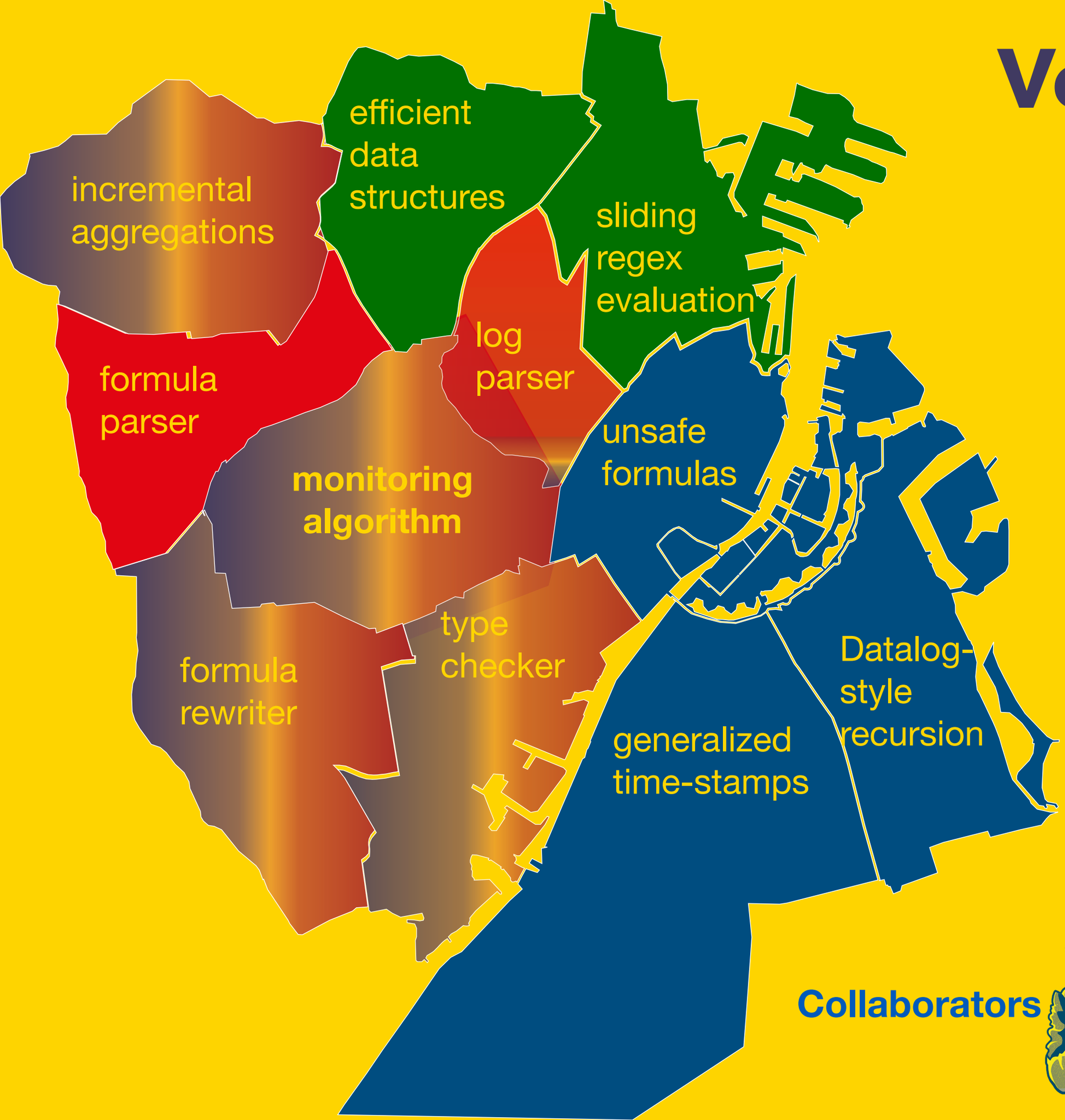
VeriMon+++

done

trustworthiness

efficiency

expressiveness



Collaborators **are welcome!**

Take Home Message

Formally verified systems
can be used as trustworthy oracles
to find bugs in unverified ones.

Take Home Message

Formally verified systems can be used as trustworthy oracles to find bugs in unverified ones.

But verified systems are easier to maintain, extend, optimize.

Take Home Message

Formally verified systems can be used as trustworthy oracles to find bugs in unverified ones.

But verified systems are easier to maintain, extend, optimize.

Instead of continuing to build, break, and fix unverified systems, we should subsume them with verified ones.

VeriMon

A Formally Verified Monitoring Tool

Dmitriy Traytel
traytel@di.ku.dk

UNIVERSITY OF COPENHAGEN