Management of Time Requirements in Component-based Systems

Yi Li[1] Tian Huat Tan[2] Marsha Chechik[1]

1. University of Toronto
2. Singapore University of Technology and Design

FM 2014 Singapore
May 14, 2014

Component-based Software Engineering

Business Goals & System Requirements

Component-based Software Engineering

Business Goals & System Requirements

Component-based Software Engineering

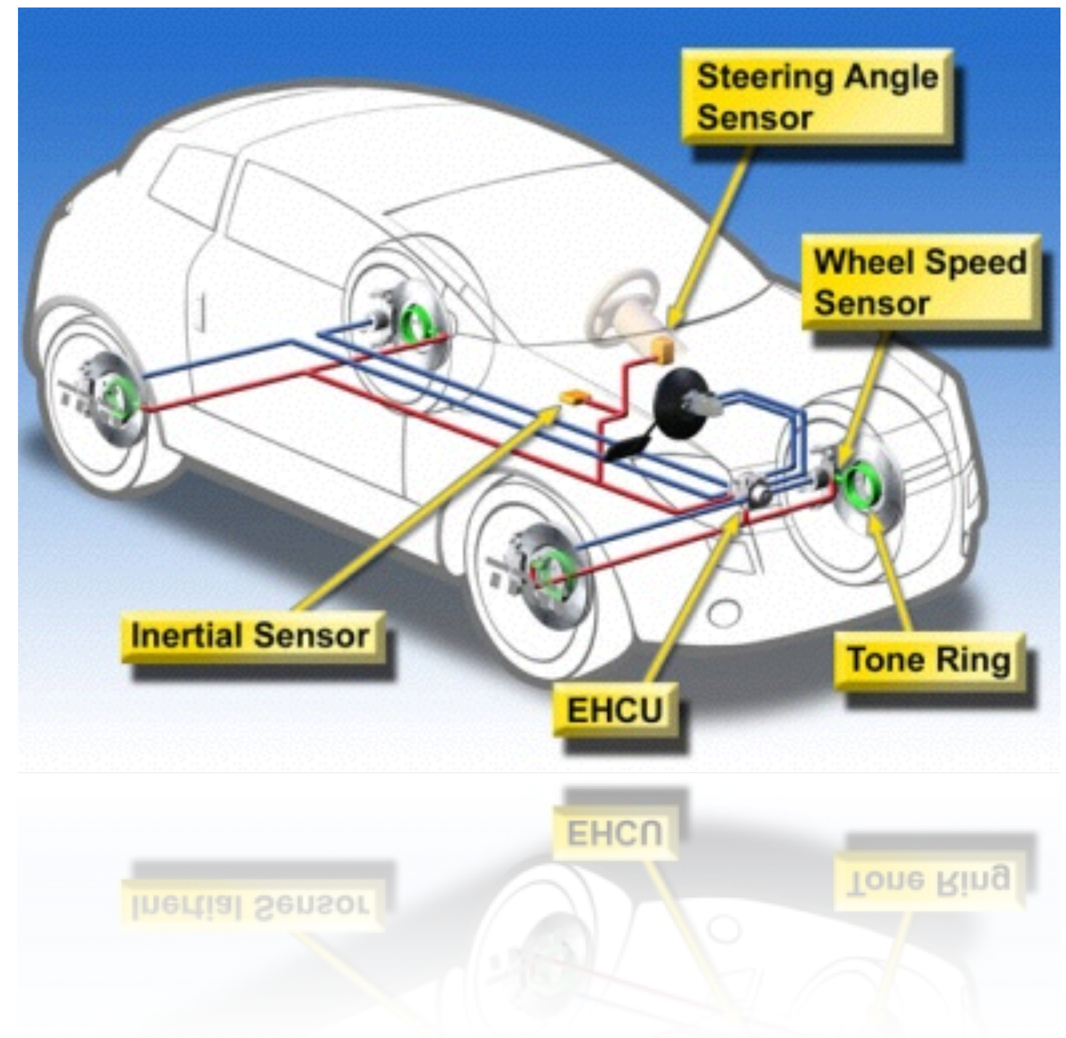
modularity, reusability, separation of concerns

Timing Requirements

Timing Requirements

Vehicle Control Systems

- *Electronic Stability Control (ESC)*
- *Anti-lock braking system (ABS)*



Timing Requirements

Vehicle Control Systems

- *Electronic Stability Control (ESC)*
- *Anti-lock braking system (ABS)*

Smart Phones

Timing Requirements

Vehicle Control Systems

- *Electronic Stability Control (ESC)*
- *Anti-lock braking system (ABS)*

Smart Phones

- *Sensors - motion tracking*

Timing Requirements

Vehicle Control Systems

- *Electronic Stability Control (ESC)*
- *Anti-lock braking system (ABS)*

Smart Phones

- *Sensors - motion tracking*

Web Service Compositions

- *Ticket Booking*
- *Stock Quotes*

Timing Requirements

Vehicle Control Systems

- *Electronic Stability Control (ESC)*
- *Anti-lock braking system (ABS)*

Smart Phones

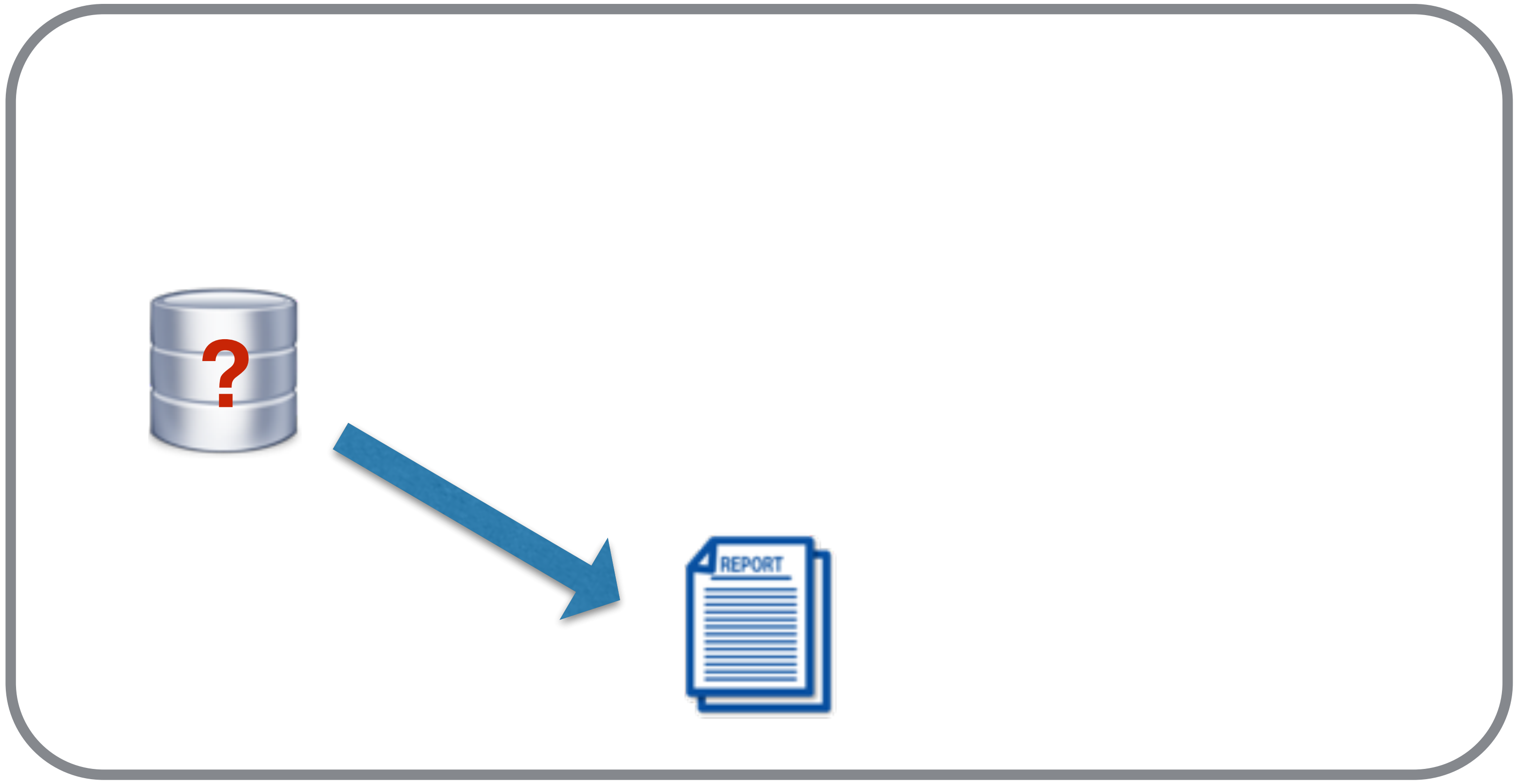
- *Sensors - motion tracking*

Web Service Compositions

- *Ticket Booking*
- *Stock Quotes*

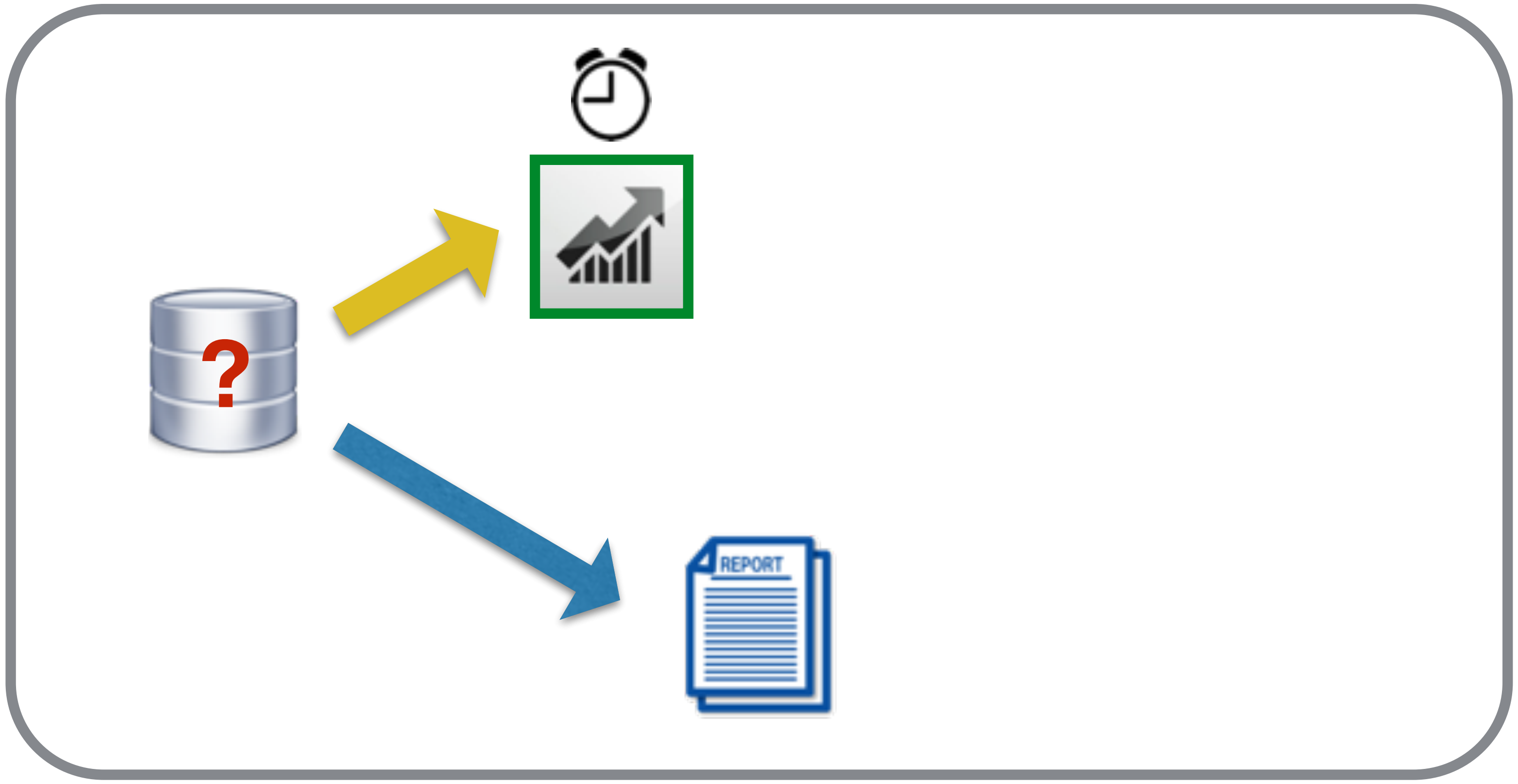
Existing Approach: LTR

Existing Approach: LTR

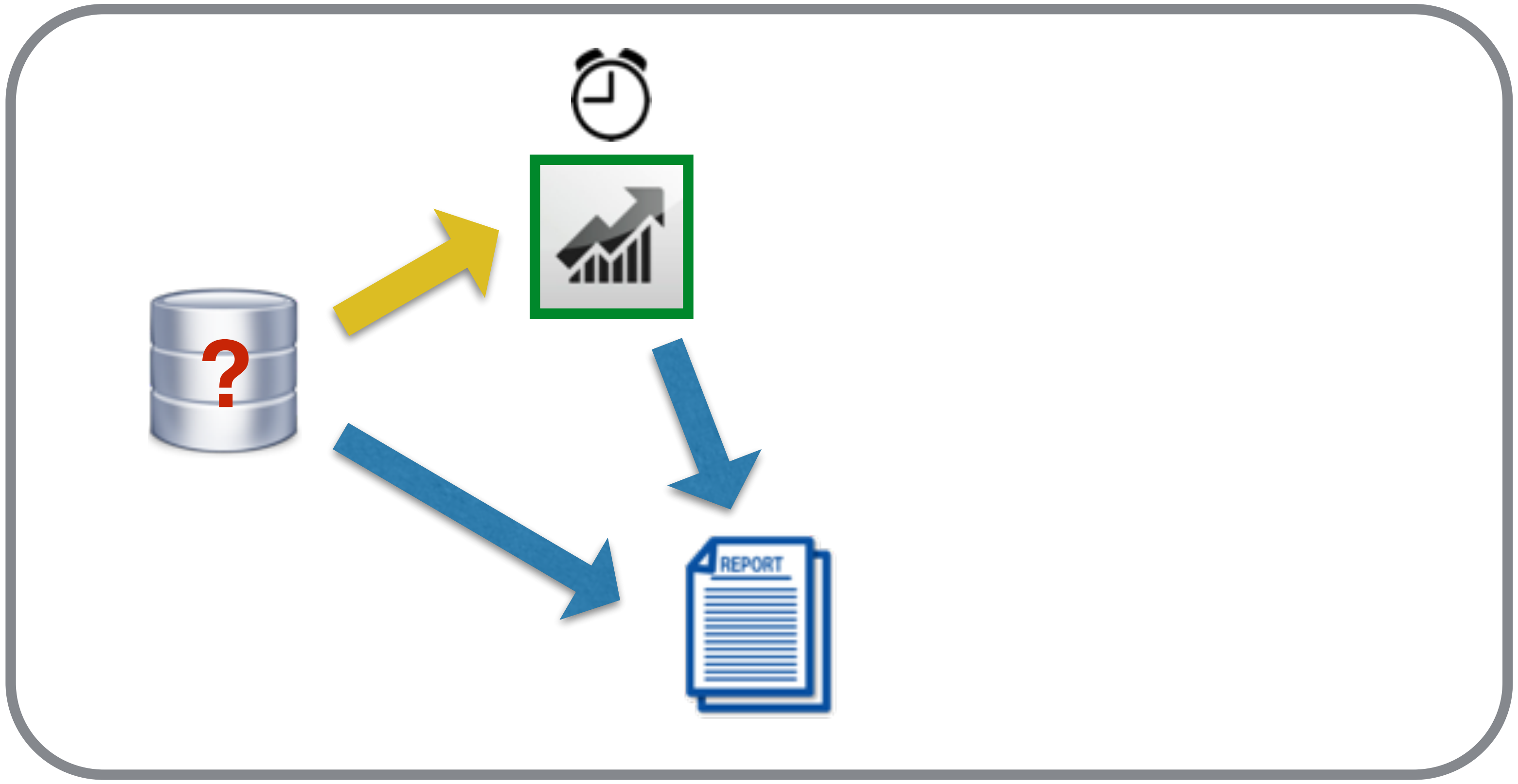


Existing Approach: LTR

Existing Approach: LTR



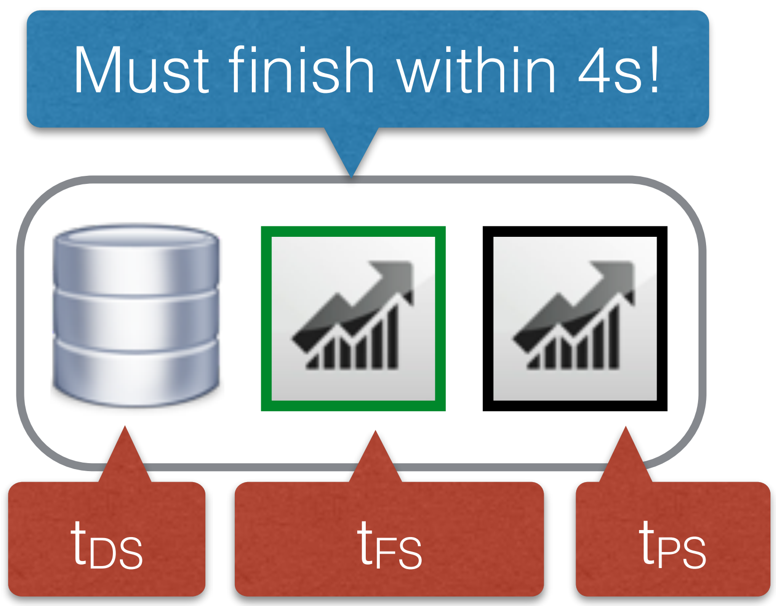
Existing Approach: LTR



Existing Approach: LTR

Existing Approach: LTR

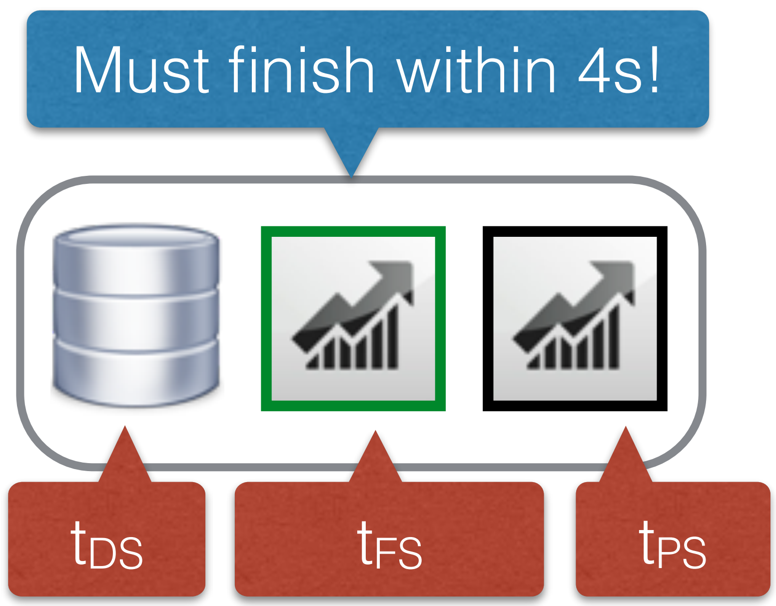
Existing Approach: LTR



Previous Work:  [ICSE'13]

- *Local Timing Requirements (LTR) synthesis*
- *Web Services - BPEL*
- *Monolithic representation*

Existing Approach: LTR



Previous Work: [ICSE'13]

- *Local Timing Requirements (LTR) synthesis*
- *Web Services - BPEL*
- *Monolithic representation*

LTR:

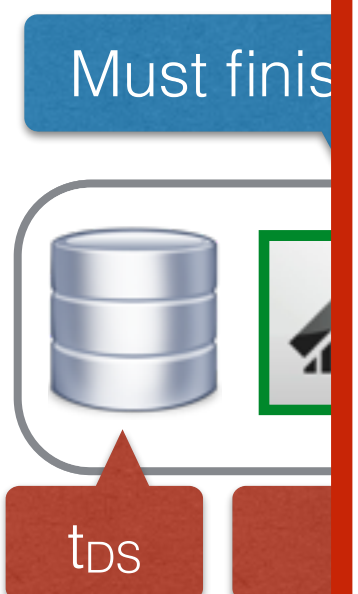
$$\neg(0 \le t_{DS} \wedge 1 \le t_{FS} \wedge 1 \le t_{PS})$$
$$\wedge((0 \le t_{DS} \wedge 0 \le t_{FS} \wedge 0 \le t_{PS}) \Rightarrow t_{DS} \le 3)$$
$$\wedge((0 \le t_{DS} \wedge 0 \le t_{FS} \le 1 \wedge 0 \le t_{PS}) \Rightarrow t_{DS} + t_{FS} \le 3)$$
$$\wedge((0 \le t_{DS} \wedge 1 \le t_{FS} \wedge 0 \le t_{PS} \le 1) \Rightarrow t_{DS} + t_{PS} \le 2)$$

Existing Approach: LTR

SE'13]

nents

ion



LTR - monolithic constraint

Pros:

\+ distills complicated composition structures into a single formula

\+ precisely captures all feasible combinations

Cons:

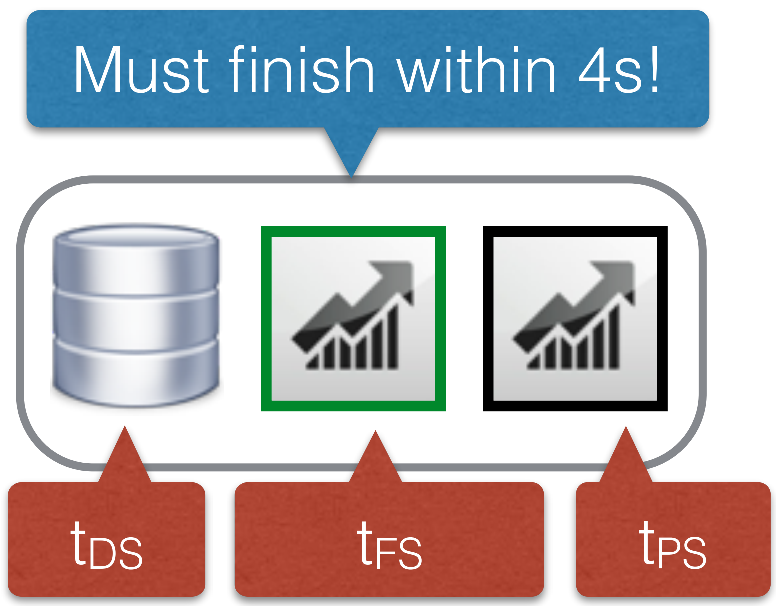
\- imposes dependencies across components

\- lacks support for localized debugging/repairing

LTR:

$$\neg(0\le t_{DS}\wedge 1\le t_{FS}\wedge 1\le t_{PS})$$
$$\wedge((0\le t_{DS}\wedge 0\le t_{FS}\wedge 0\le t_{PS})\Rightarrow t_{DS}\le 3)$$
$$\wedge((0\le t_{DS}\wedge 0\le t_{FS}\le 1\wedge 0\le t_{PS})\Rightarrow t_{DS}+t_{FS}\le 3)$$
$$\wedge((0\le t_{DS}\wedge 1\le t_{FS}\wedge 0\le t_{PS}\le 1)\Rightarrow t_{DS}+t_{PS}\le 2)$$

Existing Approach: LTR



Previous Work: [ICSE'13]

- *Local Timing Requirements (LTR) synthesis*
- *Web Services - BPEL*
- *Monolithic representation*

LTR:

$$\neg(0 \leq t_{DS} \wedge 1 \leq t_{FS} \wedge 1 \leq t_{PS})$$
$$\wedge((0 \leq t_{DS} \wedge 0 \leq t_{FS} \wedge 0 \leq t_{PS}) \Rightarrow t_{DS} \leq 3)$$
$$\wedge((0 \leq t_{DS} \wedge 0 \leq t_{FS} \leq 1 \wedge 0 \leq t_{PS}) \Rightarrow t_{DS} + t_{FS} \leq 3)$$
$$\wedge((0 \leq t_{DS} \wedge 1 \leq t_{FS} \wedge 0 \leq t_{PS} \leq 1) \Rightarrow t_{DS} + t_{PS} \leq 2)$$

uLTR:

$$(0 \leq t_{DS} < 1 \wedge 0 \leq t_{FS} < 1)$$
$$\vee(0 \leq t_{DS} < 1 \wedge 0 \leq t_{PS} < 1)$$

LTR vs. uLTR

LTR:

$$\neg(0 \le t_{DS} \wedge 1 \le t_{FS} \wedge 1 \le t_{PS})$$
$$\wedge((0 \le t_{DS} \wedge 0 \le t_{FS} \wedge 0 \le t_{PS}) \Rightarrow t_{DS} \le 3)$$
$$\wedge((0 \le t_{DS} \wedge 0 \le t_{FS} \le 1 \wedge 0 \le t_{PS}) \Rightarrow t_{DS} + t_{FS} \le 3)$$
$$\wedge((0 \le t_{DS} \wedge 1 \le t_{FS} \wedge 0 \le t_{PS} \le 1) \Rightarrow t_{DS} + t_{PS} \le 2)$$

- *Component-dependent timing requirement*
- *Linear real arithmetic*
- *Precise*
- *Monolithic*

uLTR:

$$(0 \le t_{DS} < 1 \wedge 0 \le t_{FS} < 1)$$
$$\vee(0 \le t_{DS} < 1 \wedge 0 \le t_{PS} < 1)$$

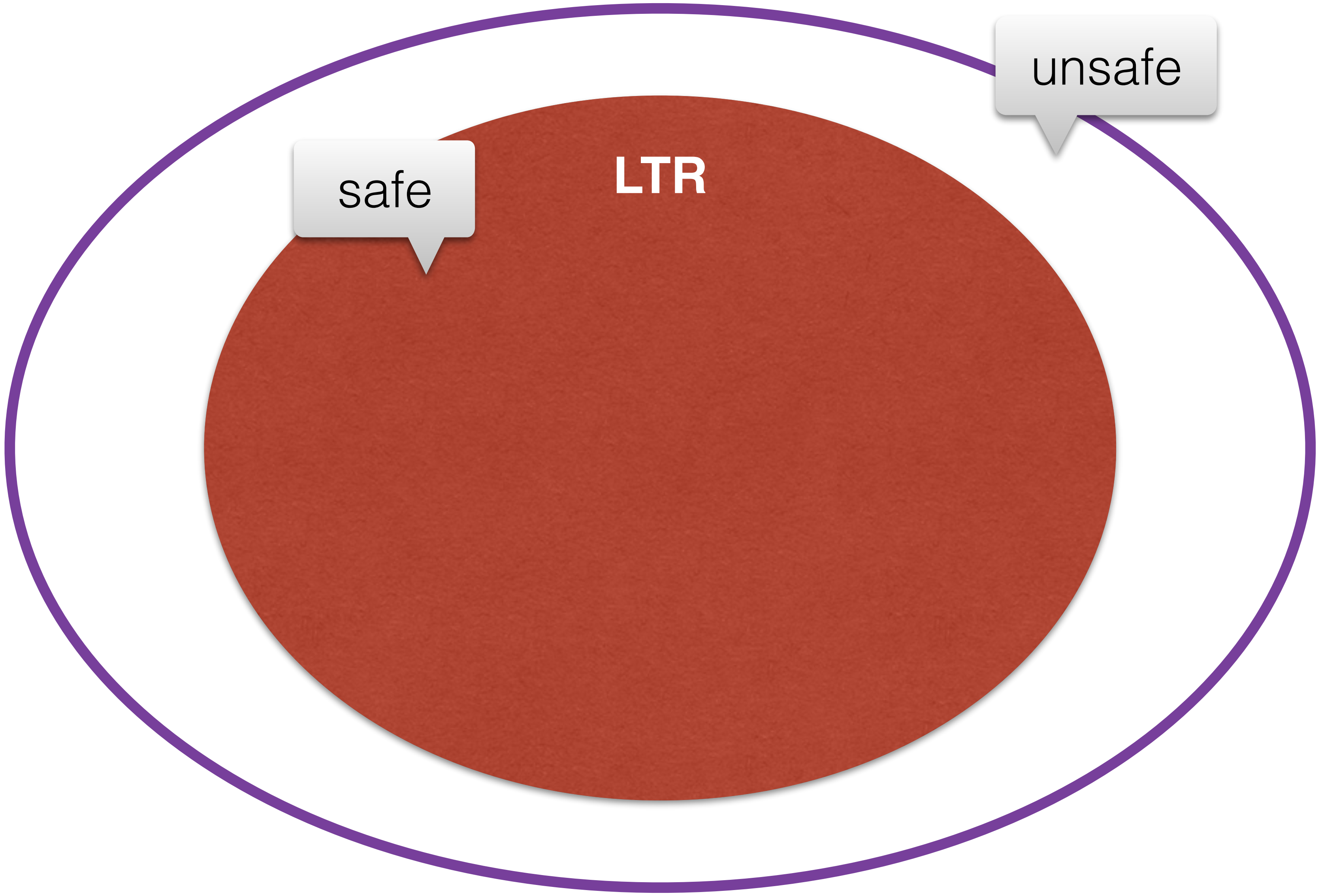
- *Component-independent under-approximated LTR*
- *Intervals*
- *Under-approximated*
- *Localized*

LTR vs. uLTR

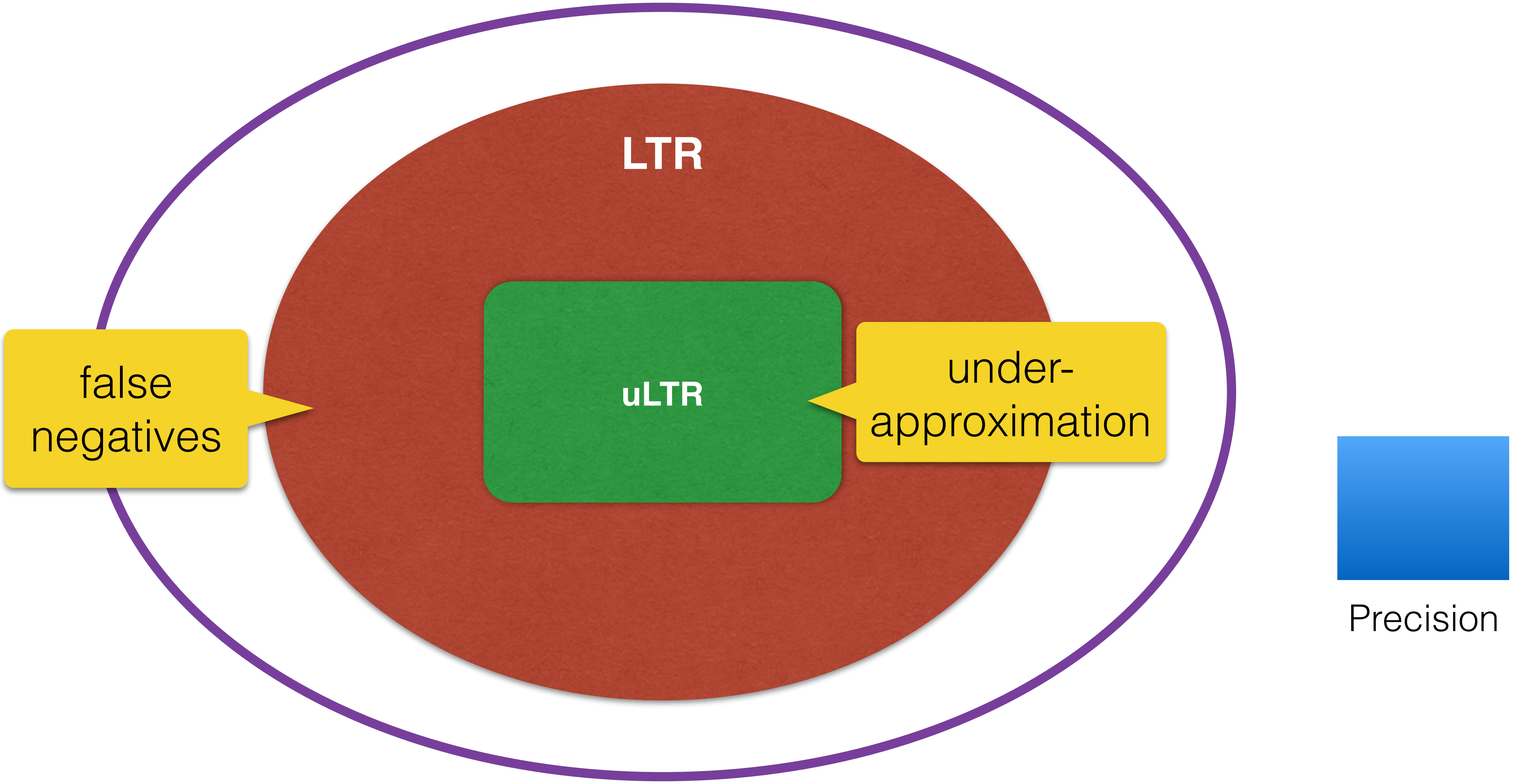
All possible timing configurations,

e.g., $t_{DS} = 1$, $t_{FS} = 0.5$, $t_{PS} = 0.8$

LTR vs. uLTR

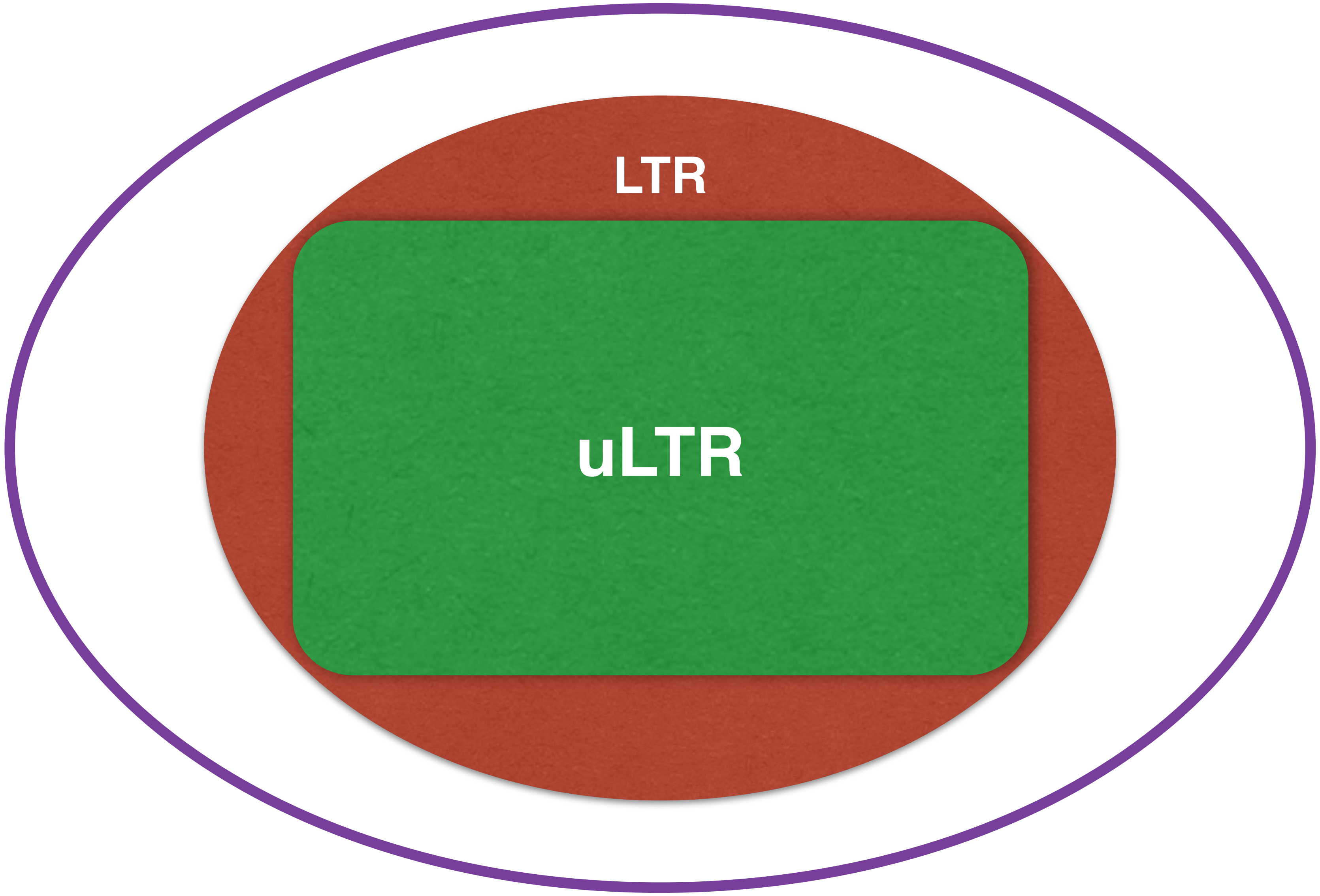


LTR vs. uLTR



$$\text{Precision}(\text{uLTR}) = \frac{\#\text{configurations satisfied by uLTR}}{\#\text{configurations satisfied by LTR}} \times 100\%$$

LTR vs. uLTR



$$\text{Precision}(\text{uLTR}) = \frac{\#\text{configurations satisfied by uLTR}}{\#\text{configurations satisfied by LTR}} \times 100\%$$

Checklist

☐ What is uLTR?

- *Component-independent under-approximated LTR*
- *Soundness: ensure timing safety*

☐ How to break up the monolithic constraint?

- *Compute uLTR from LTR*
- *Precision: preserve as many choices as possible*

☐ How can localized constraints support the management of timing requirements?

- *uLTR for component selection*
- *uLTR for runtime adaptation and recovery*

Checklist

☑ What is uLTR?

- *Component-independent under-approximated LTR*
- *Soundness: ensure timing safety*

☐ How to break up the monolithic constraint?

- *Compute uLTR from LTR*
- *Precision: preserve as many choices as possible*

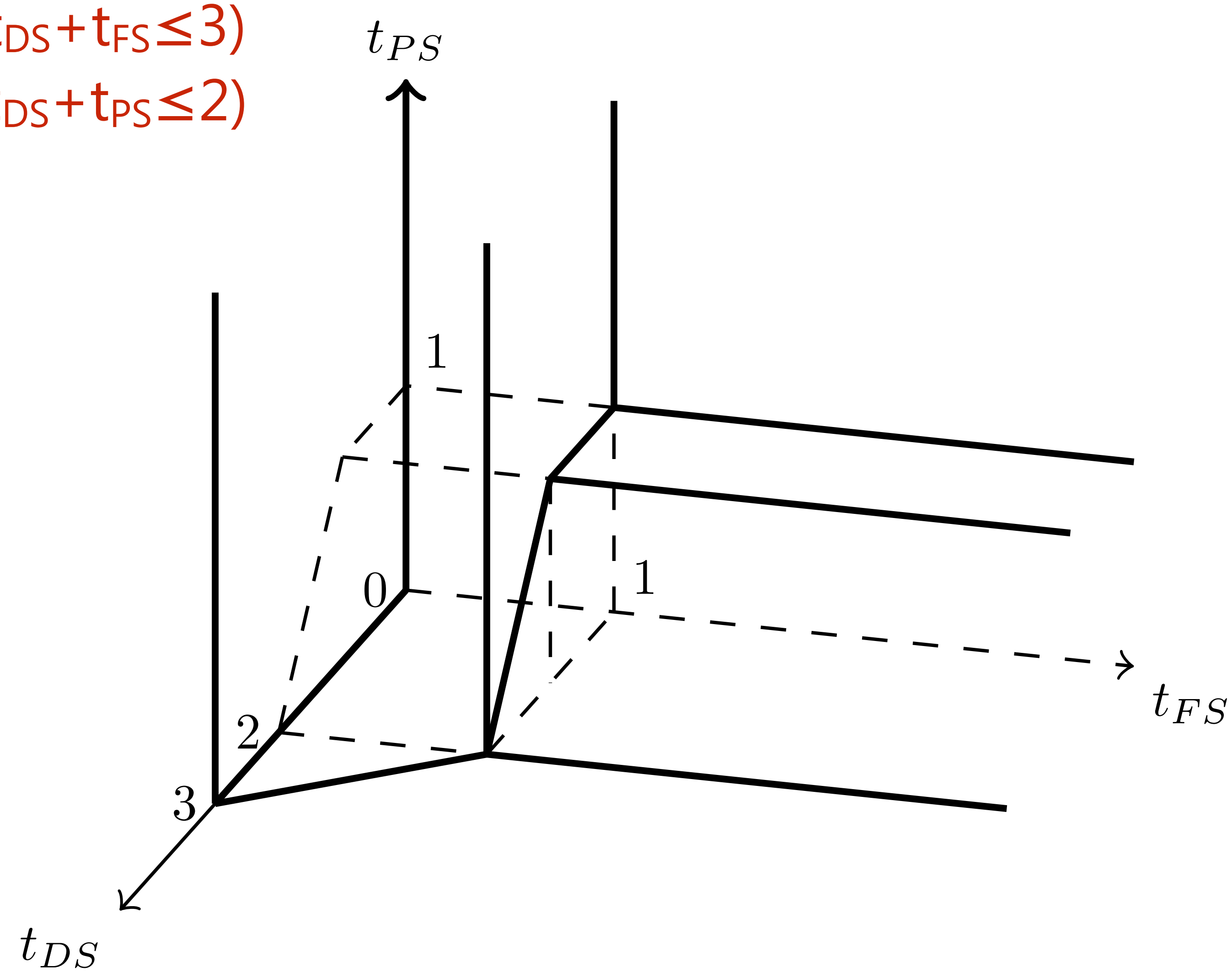
☐ How can localized constraints support the management of timing requirements?

- *uLTR for component selection*
- *uLTR for runtime adaptation and recovery*

Compute uLTR from LTR

φ:

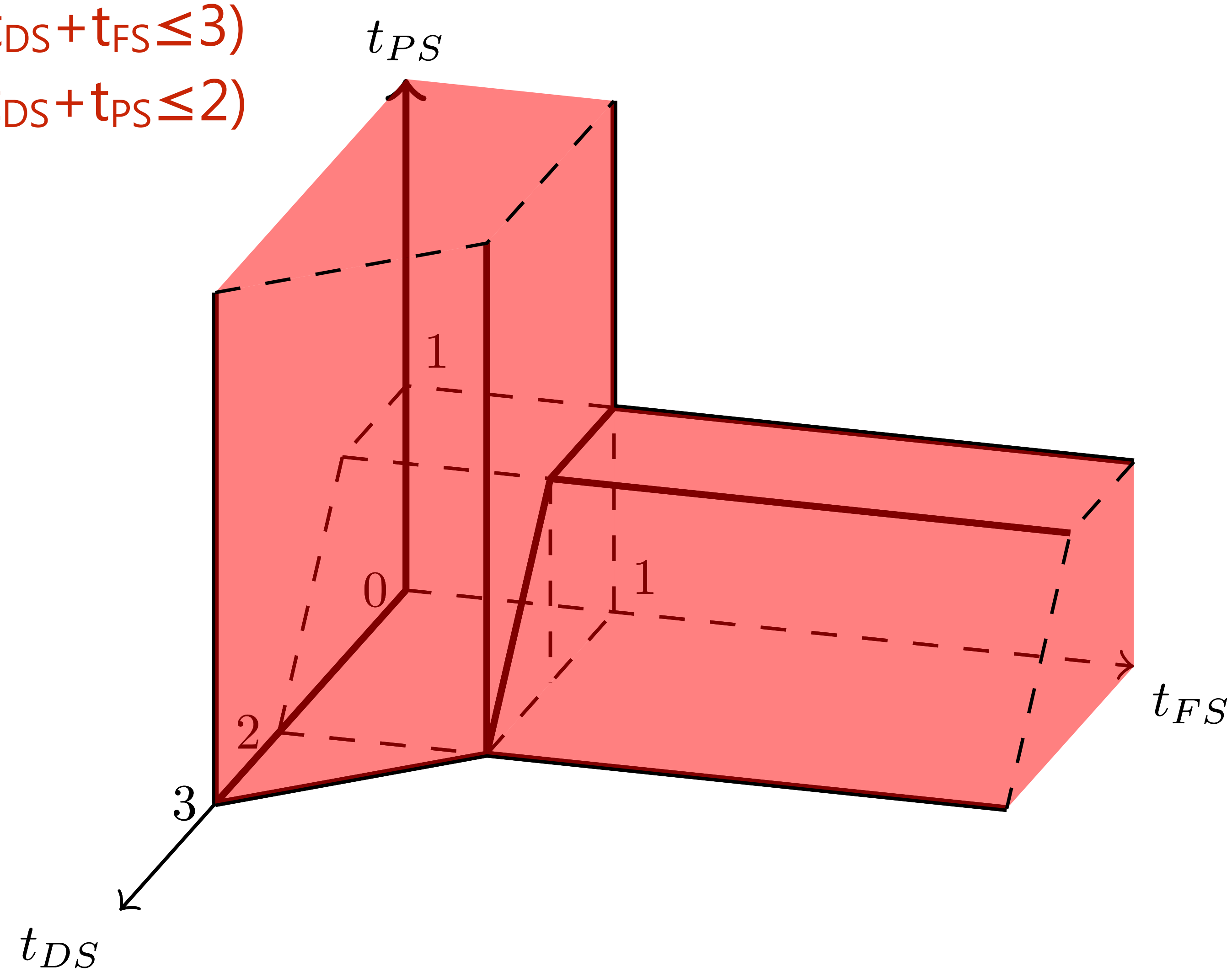
$$\neg(0 \le t_{DS} \wedge 1 \le t_{FS} \wedge 1 \le t_{PS})$$
$$\wedge((0 \le t_{DS} \wedge 0 \le t_{FS} \wedge 0 \le t_{PS}) \Rightarrow t_{DS} \le 3)$$
$$\wedge((0 \le t_{DS} \wedge 0 \le t_{FS} \le 1 \wedge 0 \le t_{PS}) \Rightarrow t_{DS} + t_{FS} \le 3)$$
$$\wedge((0 \le t_{DS} \wedge 1 \le t_{FS} \wedge 0 \le t_{PS} \le 1) \Rightarrow t_{DS} + t_{PS} \le 2)$$



Compute uLTR from LTR

φ:

$$\neg(0 \leq t_{DS} \wedge 1 \leq t_{FS} \wedge 1 \leq t_{PS})$$
$$\wedge((0 \leq t_{DS} \wedge 0 \leq t_{FS} \wedge 0 \leq t_{PS}) \Rightarrow t_{DS} \leq 3)$$
$$\wedge((0 \leq t_{DS} \wedge 0 \leq t_{FS} \leq 1 \wedge 0 \leq t_{PS}) \Rightarrow t_{DS} + t_{FS} \leq 3)$$
$$\wedge((0 \leq t_{DS} \wedge 1 \leq t_{FS} \wedge 0 \leq t_{PS} \leq 1) \Rightarrow t_{DS} + t_{PS} \leq 2)$$



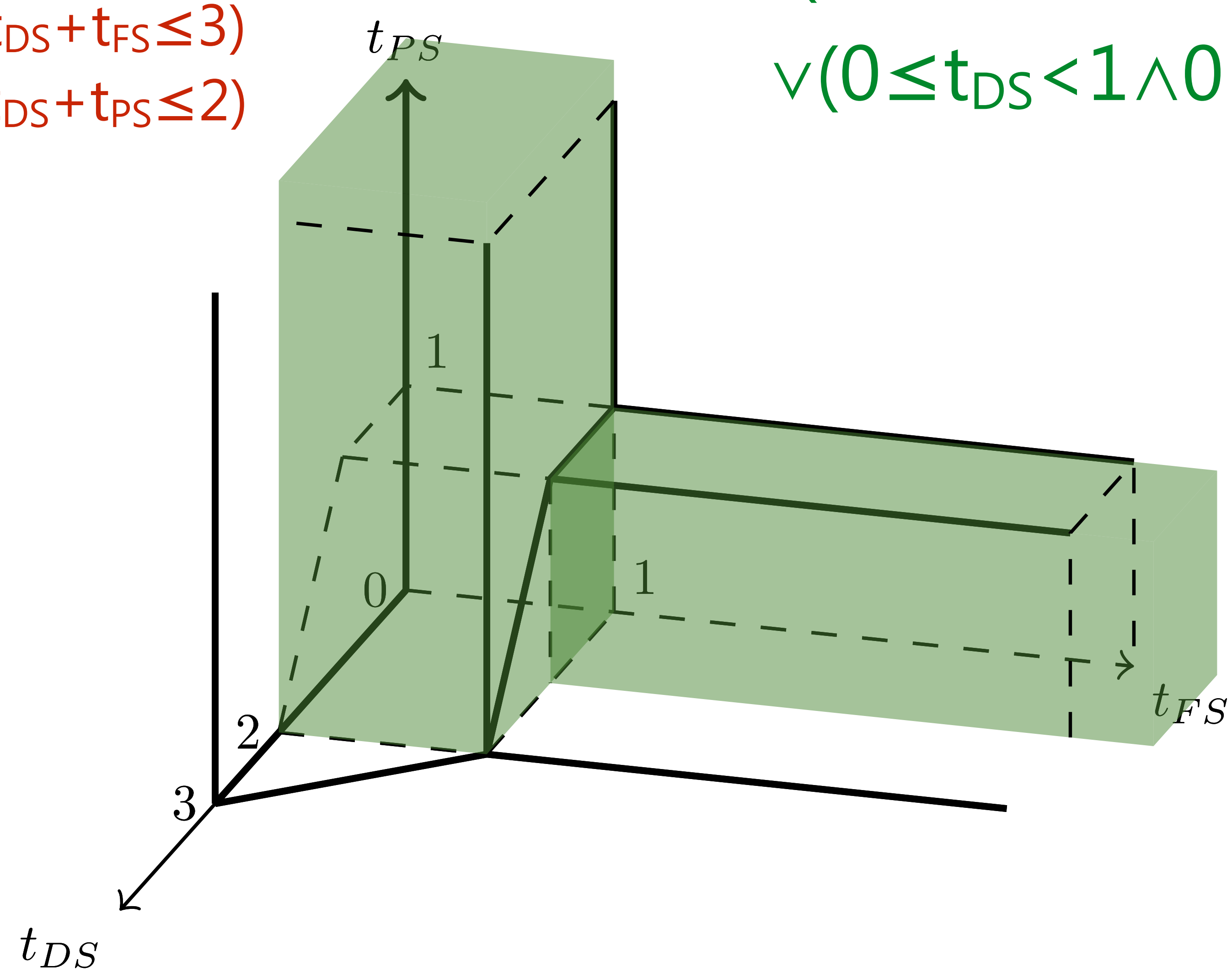
Compute uLTR from LTR

φ:

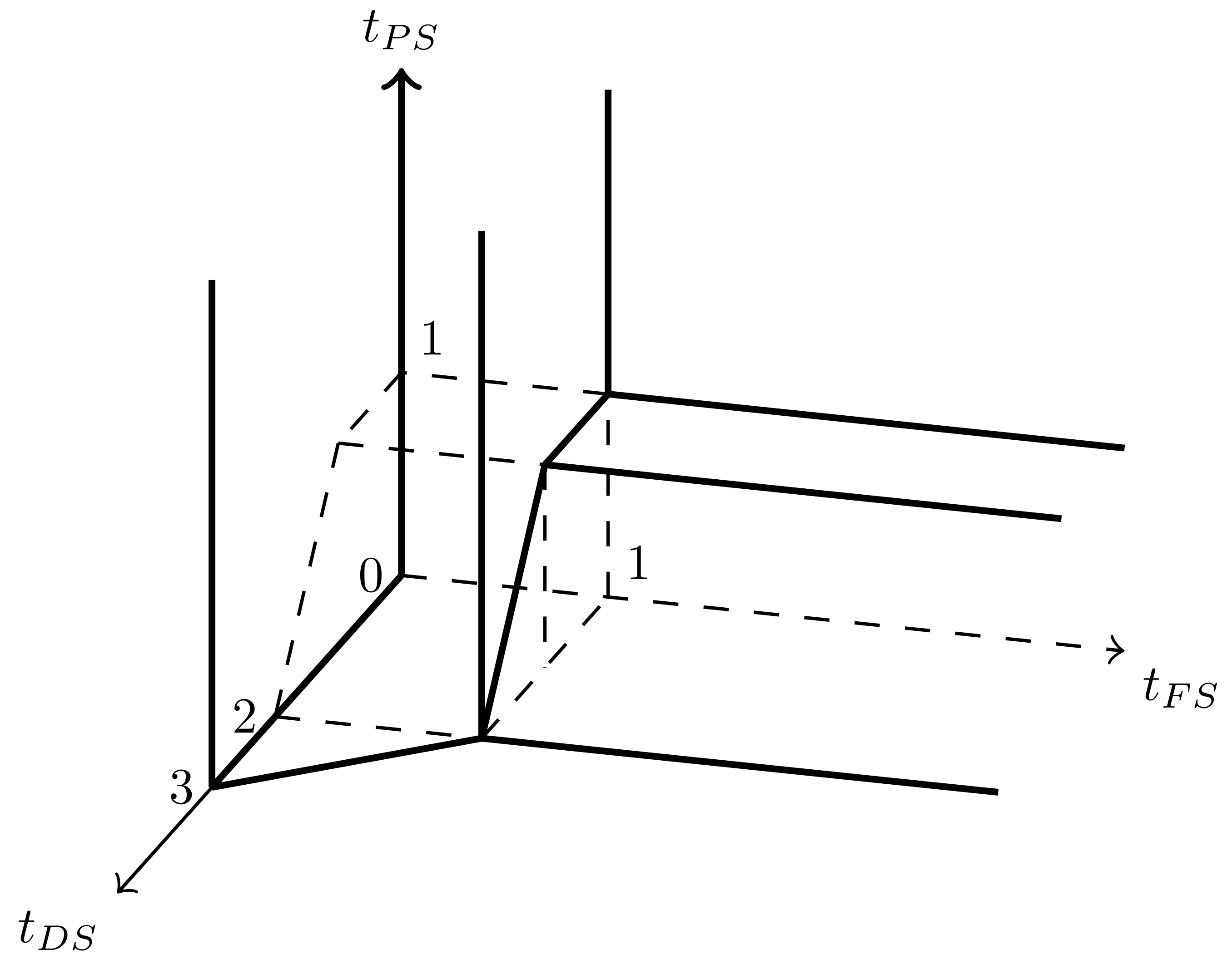
$$\neg(0 \le t_{DS} \wedge 1 \le t_{FS} \wedge 1 \le t_{PS})$$
$$\wedge((0 \le t_{DS} \wedge 0 \le t_{FS} \wedge 0 \le t_{PS}) \Rightarrow t_{DS} \le 3)$$
$$\wedge((0 \le t_{DS} \wedge 0 \le t_{FS} \le 1 \wedge 0 \le t_{PS}) \Rightarrow t_{DS} + t_{FS} \le 3)$$
$$\wedge((0 \le t_{DS} \wedge 1 \le t_{FS} \wedge 0 \le t_{PS} \le 1) \Rightarrow t_{DS} + t_{PS} \le 2)$$

B:

$$(0 \le t_{DS} < 1 \wedge 0 \le t_{FS} < 1)$$
$$\vee(0 \le t_{DS} < 1 \wedge 0 \le t_{PS} < 1)$$

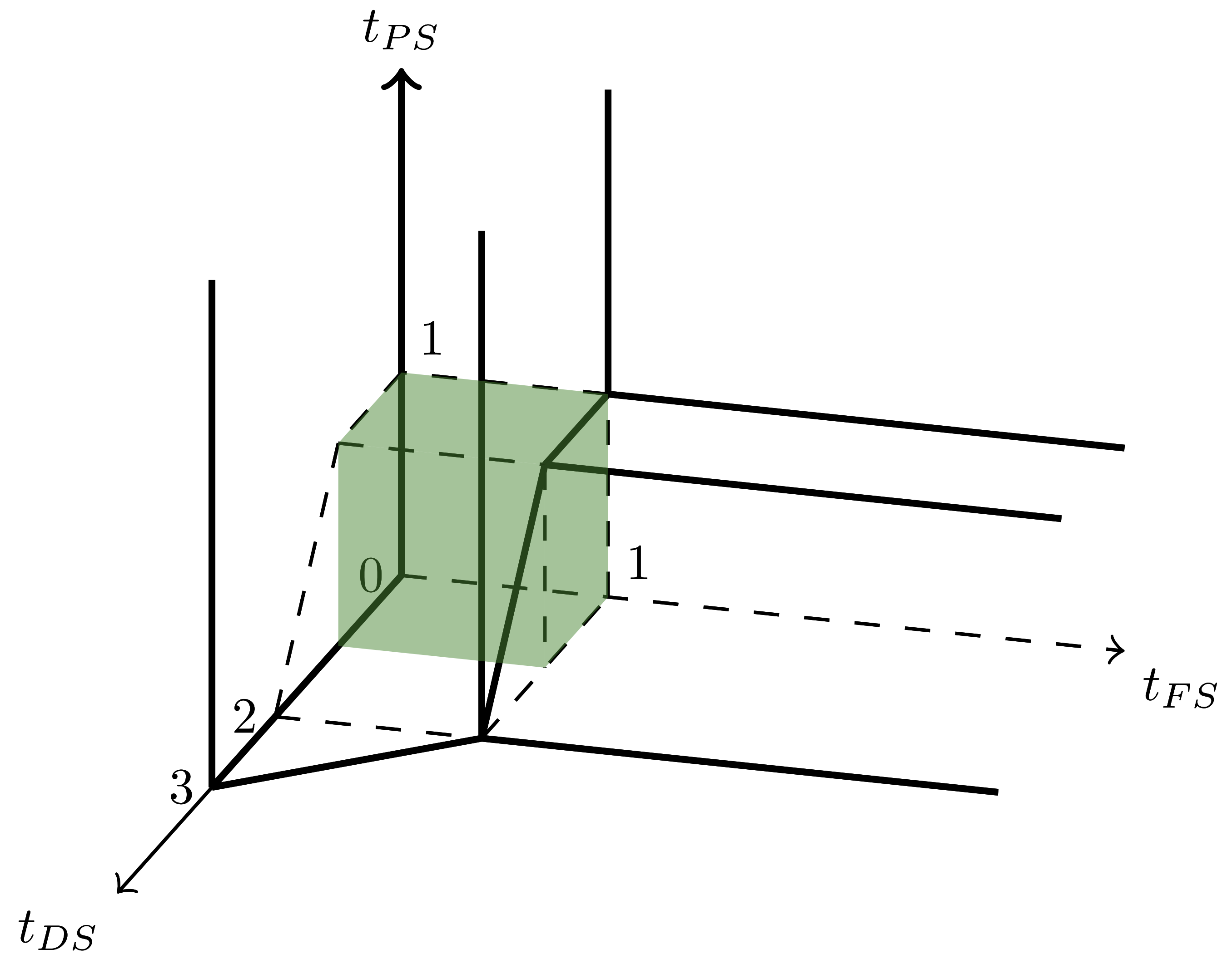


Compute uLTR from LTR



Compute uLTR from LTR

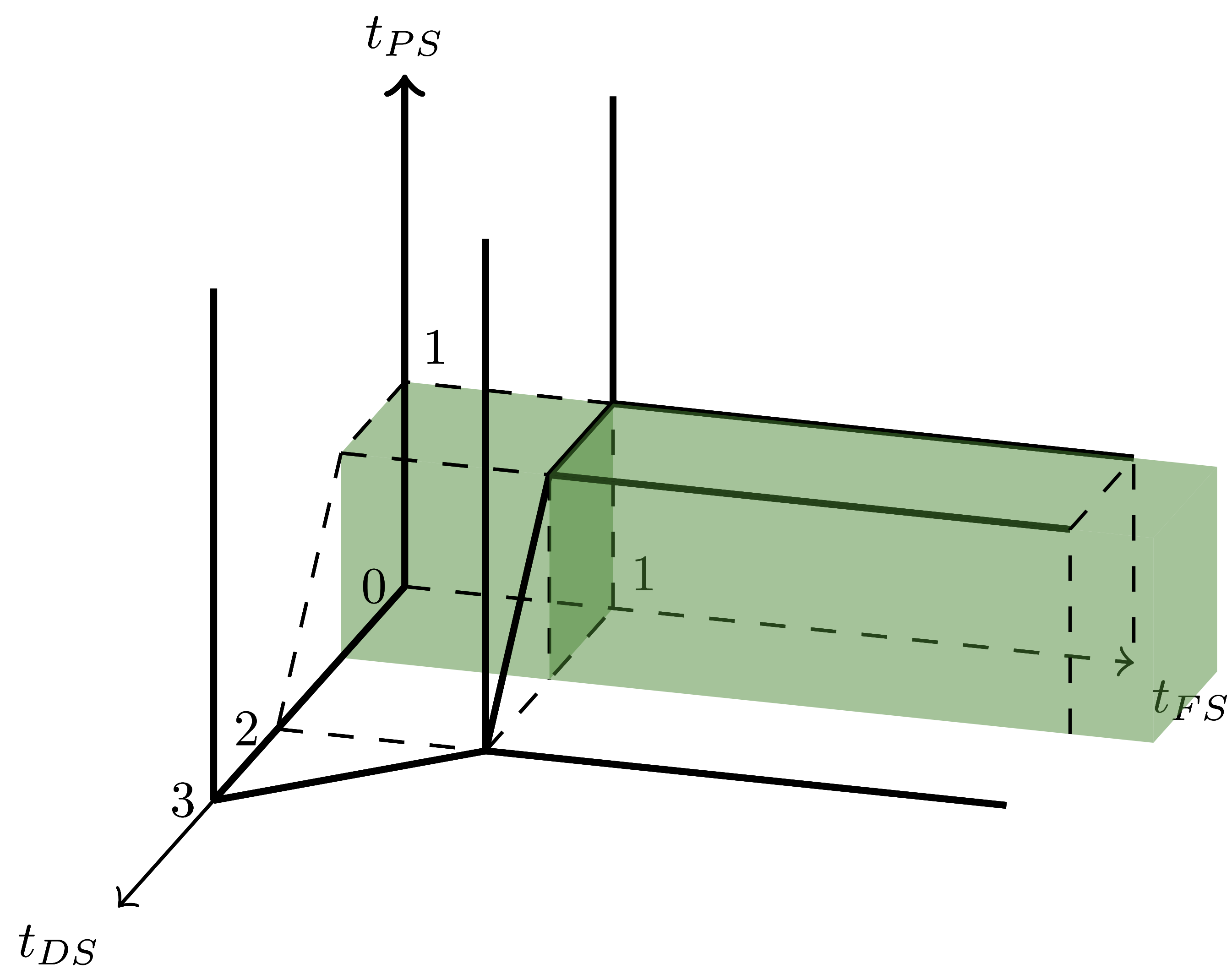
B_1= MaxCube(φ)



Compute uLTR from LTR

B_1= MaxCube(φ)

InfCube(φ,B_1)

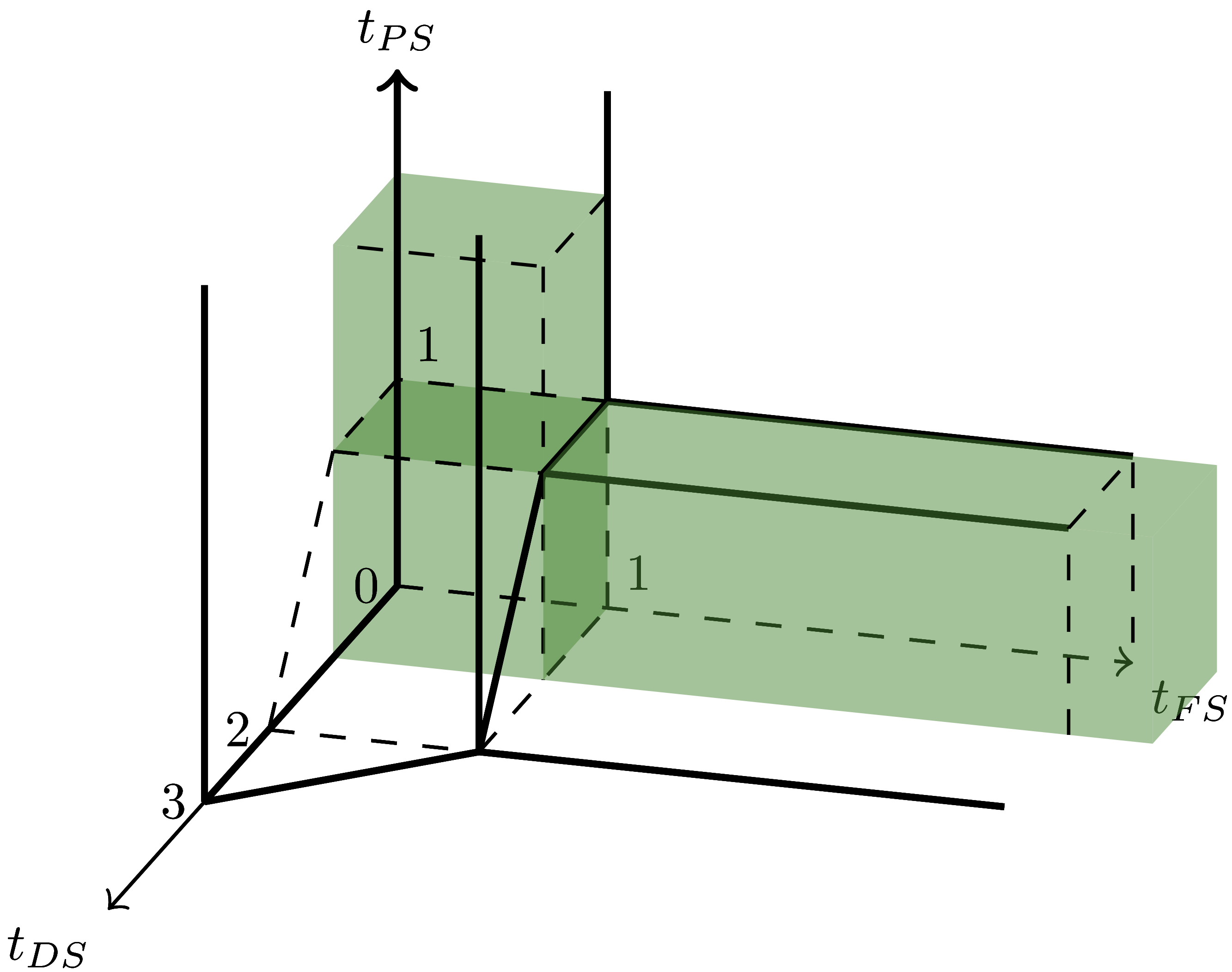


Compute uLTR from LTR

B₁= MaxCube(φ)

InfCube(φ,B₁)

B₂= MaxCube(φ)



Compute uLTR from LTR

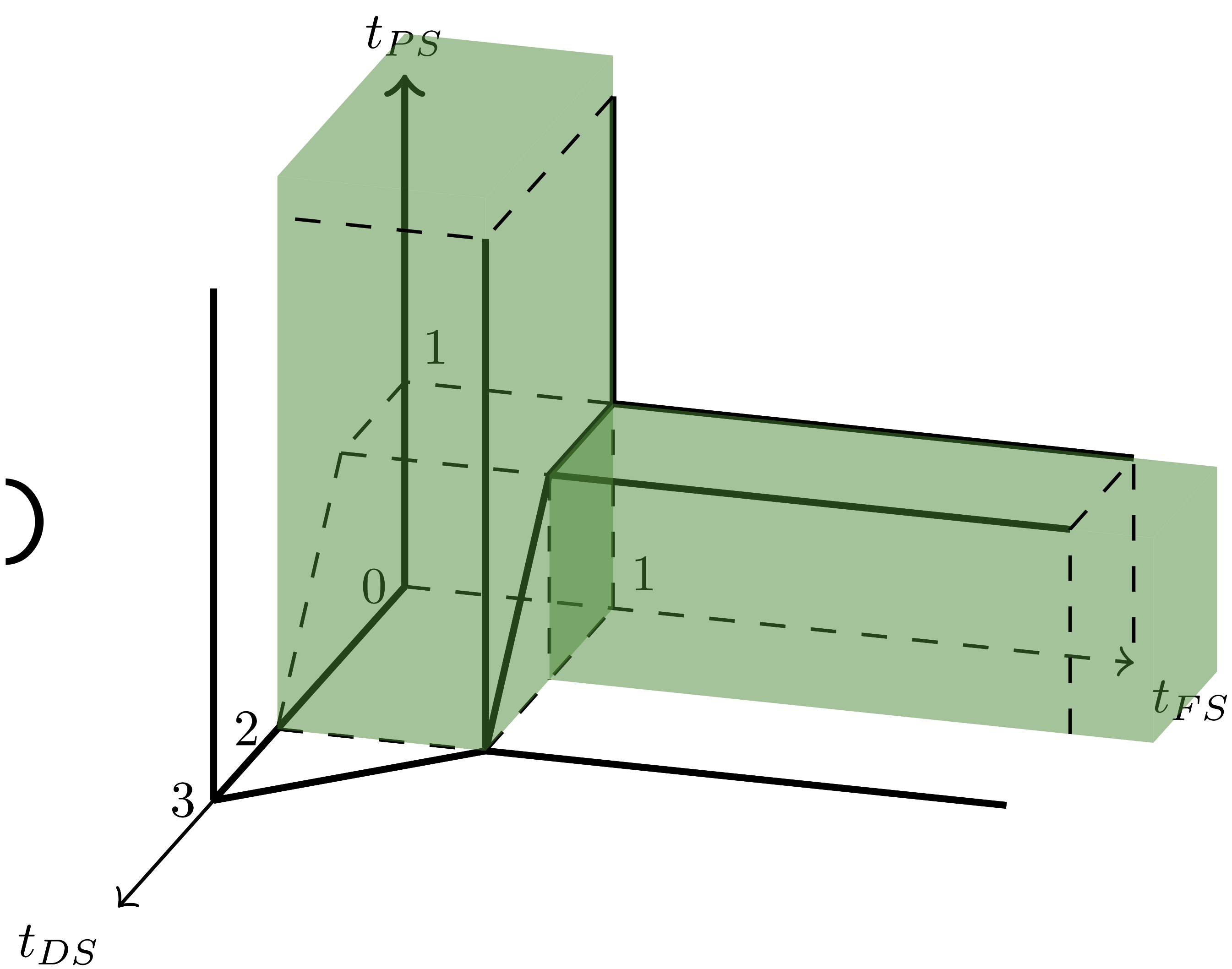
B_1= MaxCube(φ)

InfCube(φ,B_1)

B_2= MaxCube(φ)

…

B=Merge(B_1,…,B_i)



Compute uLTR from LTR

B_1= MaxCube(φ)

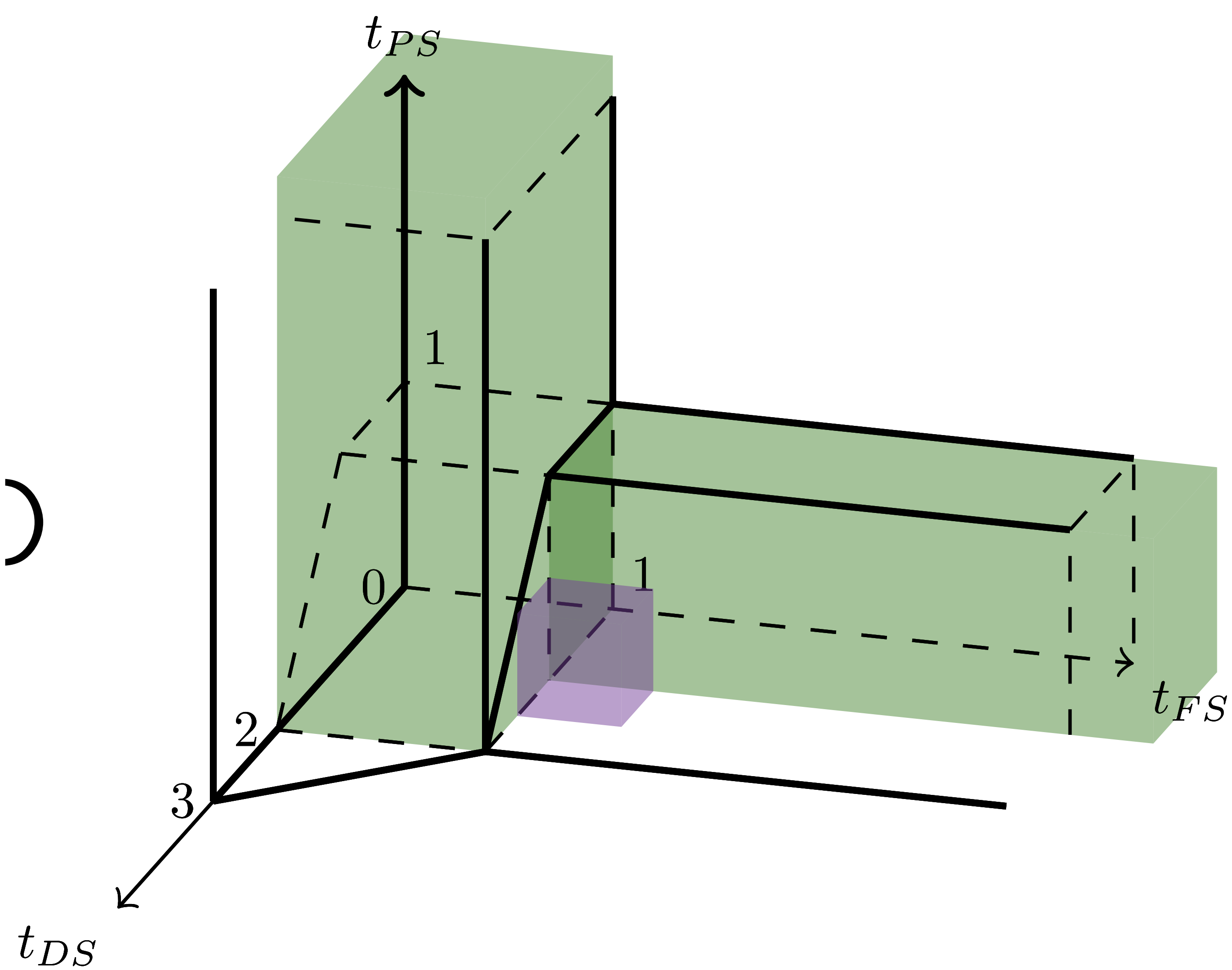
InfCube(φ,B_1)

B_2= MaxCube(φ)

…

B=Merge(B_1,…,B_i)

if (h(B_i)<ω)
   return;



Compute uLTR from LTR

B_1= MaxCube(φ)

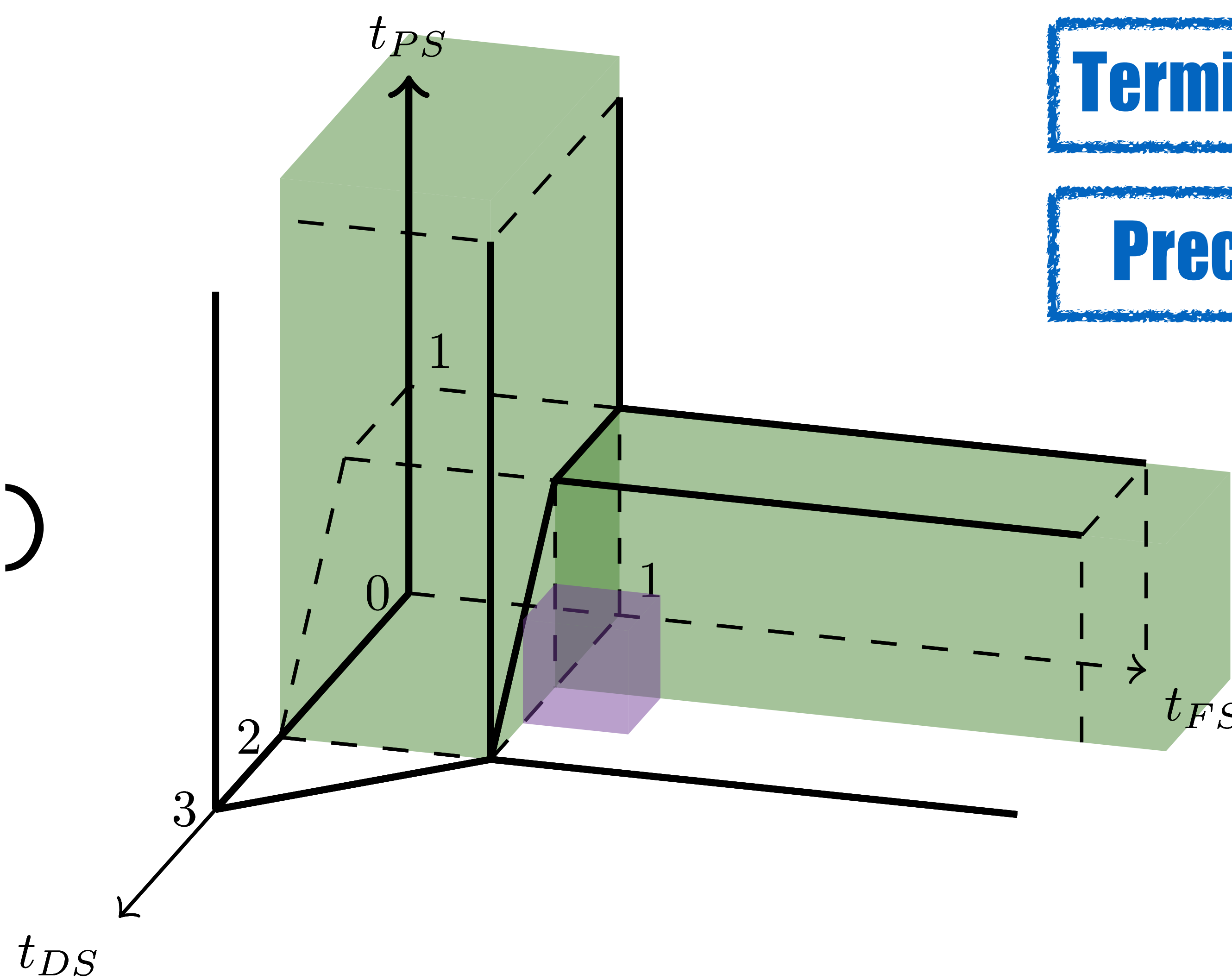
InfCube(φ,B_1)

B_2= MaxCube(φ)

…

B=Merge(B_1,…,B_i)

```
if (h(Bi)<ω)
   return;
```



Soundness

Termination

Precision

SMT Encodings

```
MaxCube(φ) //return the hypercube in φ with maximum volume
```

```
InfCube(φ,B) //relax in one direction if possible
```

SMT Encodings

MaxCube(φ) //return the hypercube in φ with maximum volume

// sample arbitrary hyper-rectangle

$$\theta \triangleq \forall Vars(\varphi) \cdot \left(\left(\bigwedge_{v_i \in Vars(\varphi)} l_i \le v_i \le u_i\right) \Rightarrow \varphi\right)$$

InfCube(φ,B) //relax in one direction if possible

SMT Encodings

```
MaxCube(φ) //return the hypercube in φ with maximum volume
    // sample arbitrary hyper-rectangle
```

$$\theta \triangleq \forall Vars(\varphi) \cdot ((\bigwedge_{v_i \in Vars(\varphi)} l_i \leq v_i \leq u_i) \Rightarrow \varphi)$$

```
    // sample maximal hyper-cube
```

$$\textsc{Optimize}(\theta \wedge (\bigwedge_{v_i \in Vars(\varphi)} (u_i - l_i = h)), h)$$

```
InfCube(φ,B) //relax in one direction if possible
```

SMT Encodings

MaxCube(φ) //return the hypercube in φ with maximum volume

// sample arbitrary hyper-rectangle

Symbolic Optimization

[POPL'14]

$$) \cdot ((\bigwedge_{v_i \in Vars(\varphi)} l_i \leq v_i \leq u_i) \Rightarrow \varphi)$$

imal hyper-cube

$$\textsc{Optimize}(\theta \wedge (\bigwedge_{v_i \in Vars(\varphi)} (u_i - l_i = h)), h)$$

InfCube(φ,B) //relax in one direction if possible

SMT Encodings

MaxCube(φ) //return the hypercube in φ with maximum volume

// sample arbitrary hyper-rectangle

$$\theta \triangleq \forall Vars(\varphi) \cdot ((\bigwedge_{v_i \in Vars(\varphi)} l_i \leq v_i \leq u_i) \Rightarrow \varphi)$$

// sample maximal hyper-cube

$$\textsc{Optimize}(\theta \wedge (\bigwedge_{v_i \in Vars(\varphi)} (u_i - l_i = h)), h)$$

InfCube(φ,B) //relax in one direction if possible

$\textsc{unSAT?}\ (\neg(B[l_i/\infty] \Rightarrow \varphi))$ // relax lower bound

$\textsc{unSAT?}\ (\neg(B[u_i/\infty] \Rightarrow \varphi))$ // relax upper bound

SMT Encodings

MaxCube(φ) //return the hypercube in φ with maximum volume

// sample arbitrary hyper-rectangle

$$\theta \triangleq \forall Vars(\varphi) \cdot ((\bigwedge_{v_i \in Vars(\varphi)} l_i \leq v_i \leq u_i) \Rightarrow \varphi)$$

// sample maximal hyper-cube

$$\textsc{Optimize}(\theta \wedge (\bigwedge_{v_i \in Vars(\varphi)} (u_i - l_i = h)), h)$$

InfCube(φ,B) //relax in one direction if possible

$\textsc{unSAT?}\ (\neg(B[l_i/\infty] \Rightarrow \varphi))$ // relax lower bound

$\textsc{unSAT?}\ (\neg(B[u_i/\infty] \Rightarrow \varphi))$ // relax upper bound

// heights of sampled hyper-cubes form a non-increasing sequence

Checklist

☑ What is uLTR?

- *Component-independent under-approximated LTR*
- *Soundness: ensure timing safety*

☐ How to break up the monolithic constraint?

- *Compute uLTR from LTR*
- *Precision: preserve as many choices as possible*

☐ How can localized constraints support the management of timing requirements?

- *uLTR for component selection*
- *uLTR for runtime adaptation and recovery*

Checklist

☑ What is uLTR?

- *Component-independent under-approximated LTR*
- *Soundness: ensure timing safety*

☑ How to break up the monolithic constraint?

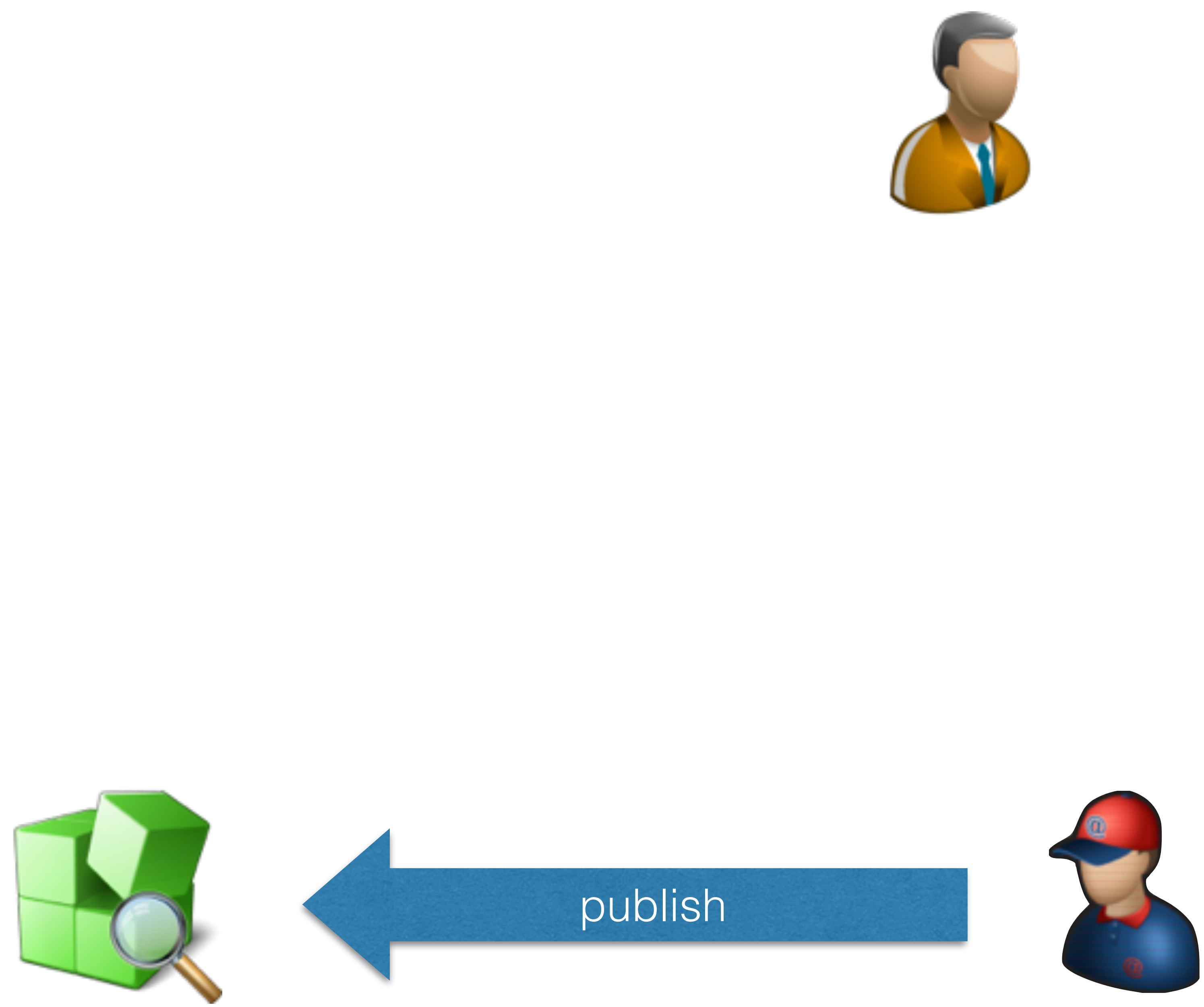
- *Compute uLTR from LTR*
- *Precision: preserve as many choices as possible*

☐ How can localized constraints support the management of timing requirements?

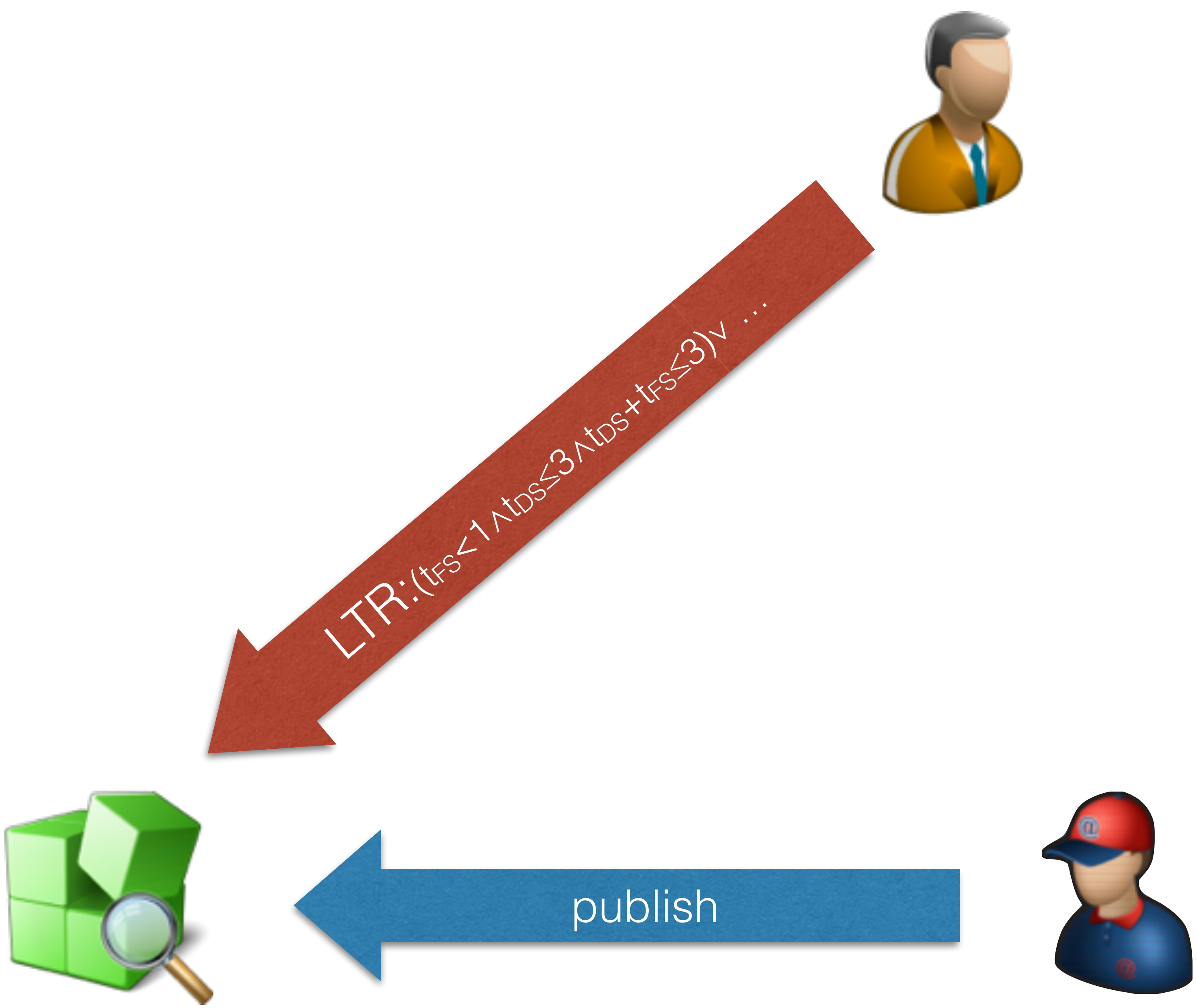
- *uLTR for component selection*
- *uLTR for runtime adaptation and recovery*

uLTR for component selection

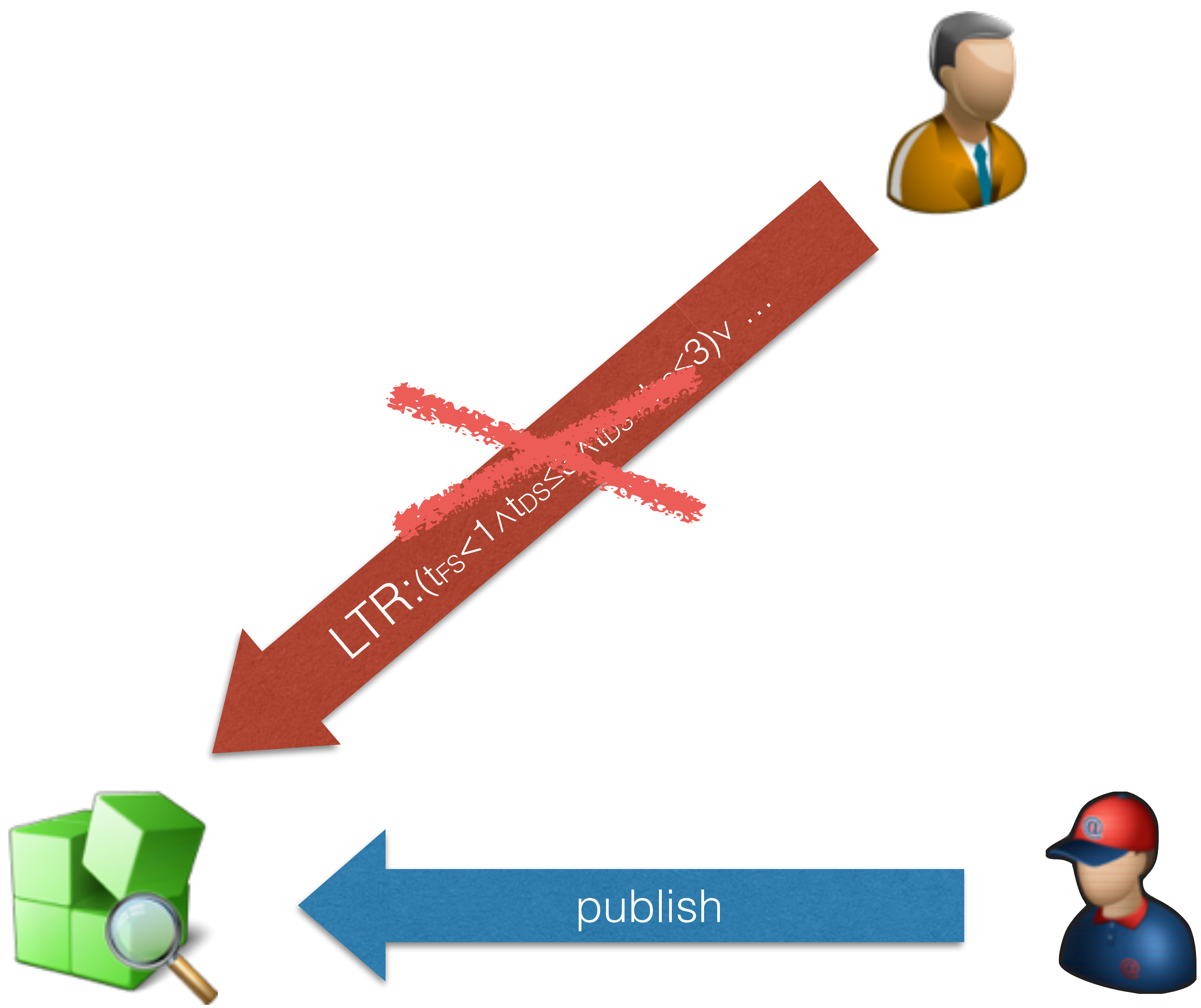
uLTR for component selection



uLTR for component selection

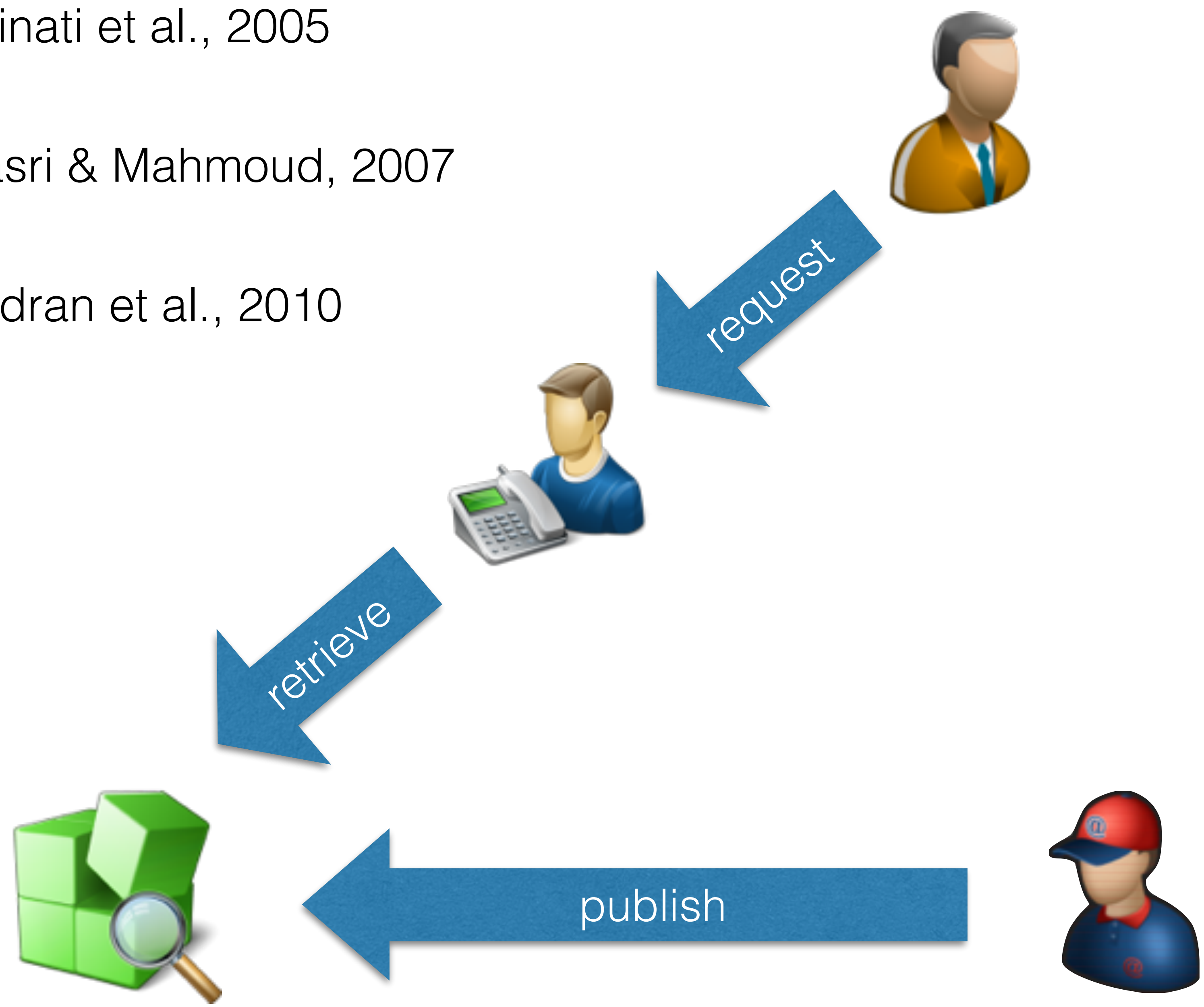


uLTR for component selection

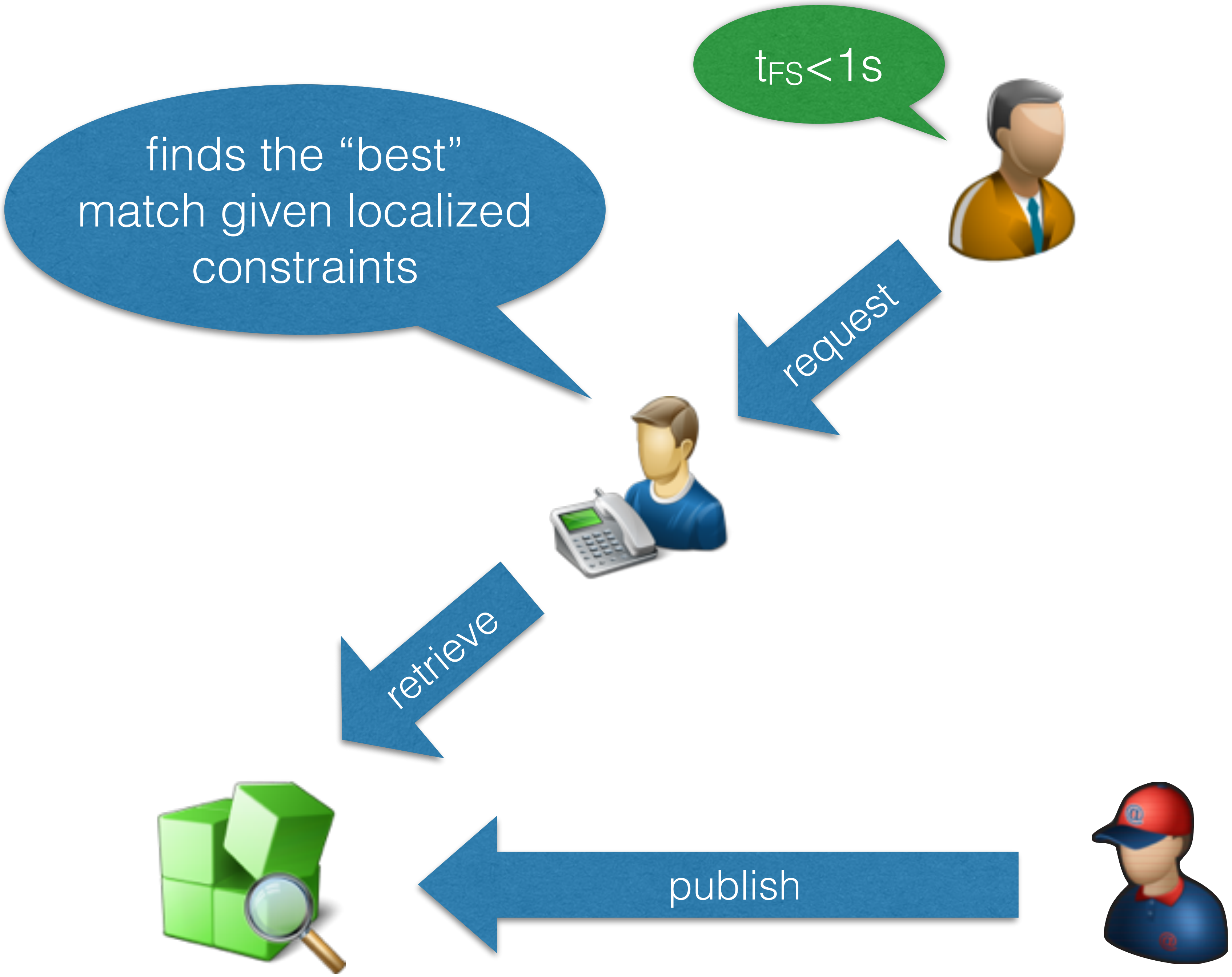


uLTR for component selection

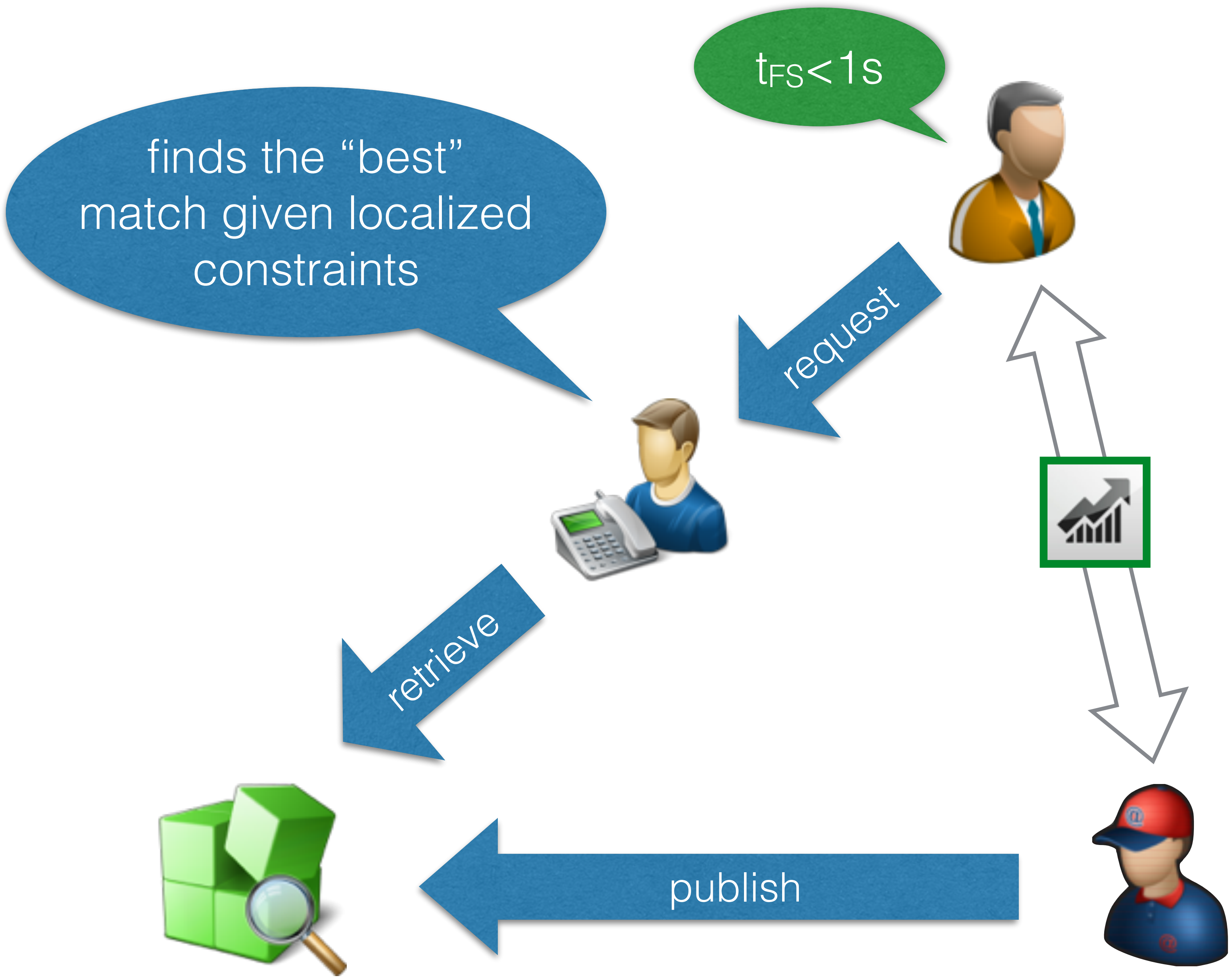
- Carminati et al., 2005
- Al-Masri & Mahmoud, 2007
- Rajendran et al., 2010



uLTR for component selection

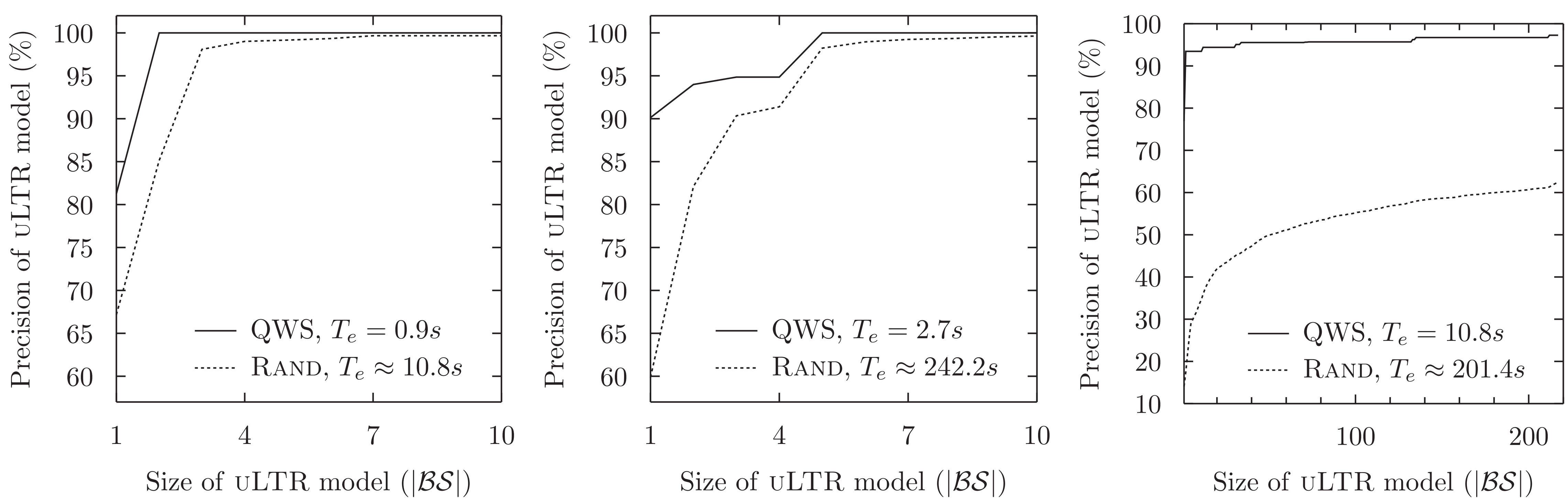


uLTR for component selection



uLTR for component selection

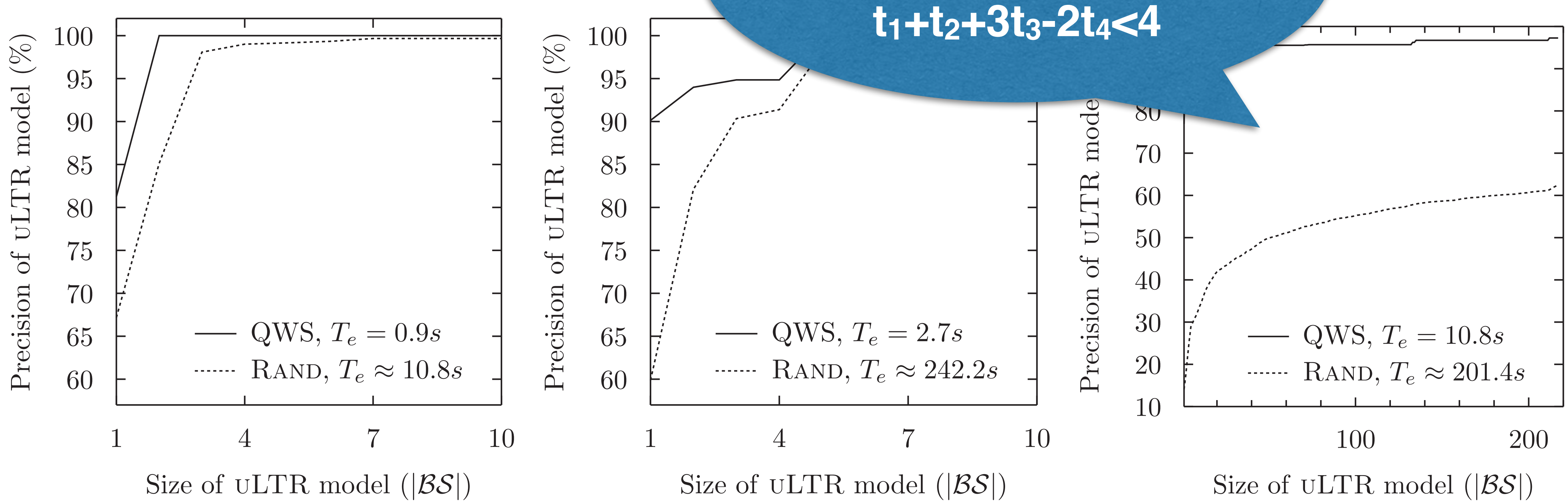
- *Real-world Web Service data: QWS dataset*
- *Case studies: online booking service, …*
- *Evaluate the percentage of false-negatives (precision) w.r.t. size of the uLTR model*



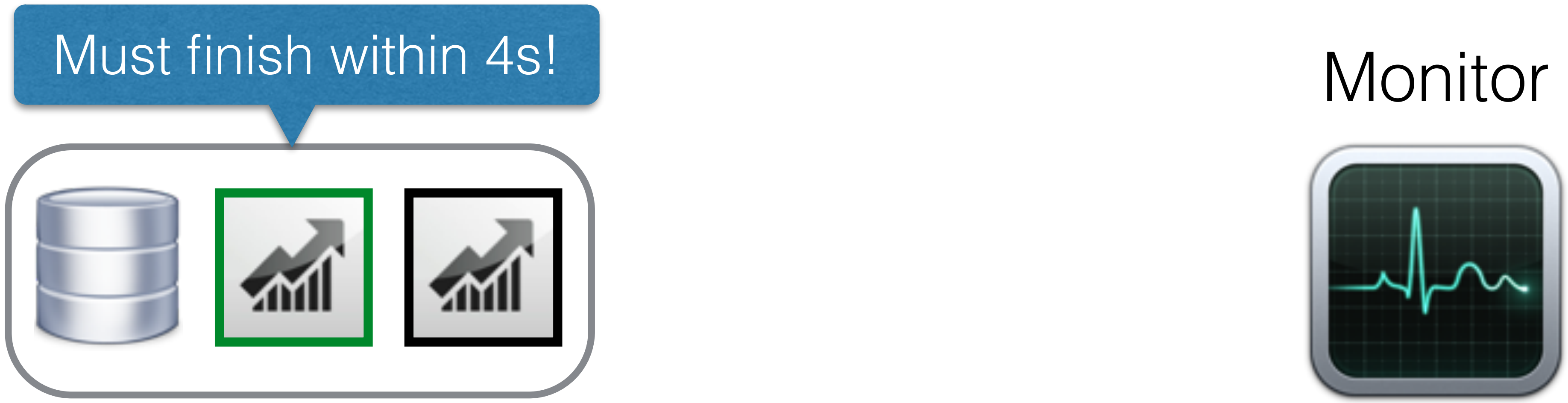
uLTR for component selection

- *Real-world Web Service data: QWS dataset*
- *Case studies: online booking service, …*
- *Evaluate the percentage of f ... ves (precision) w.r.t. size of the uLTR m*

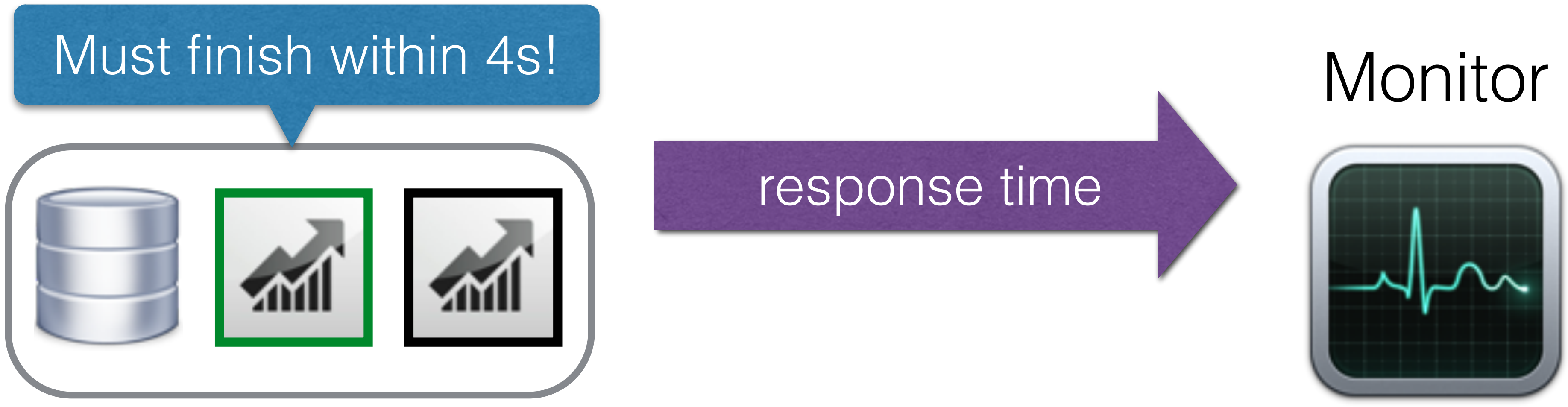
Strong dependency in the original LTR:
$t_1+t_2+3t_3-2t_4<4$



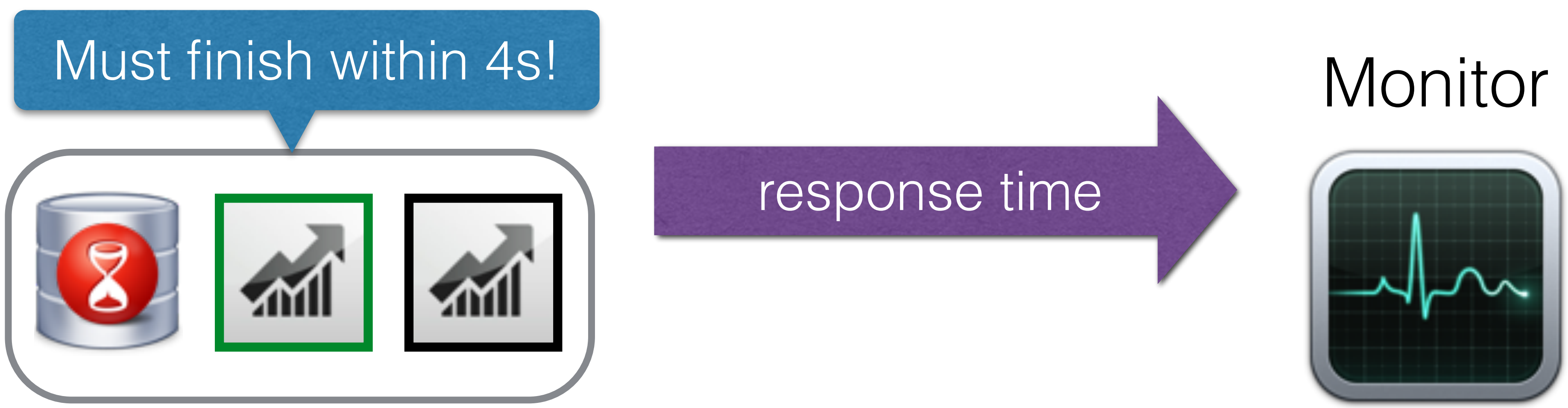
uLTR for runtime adaptation and recovery



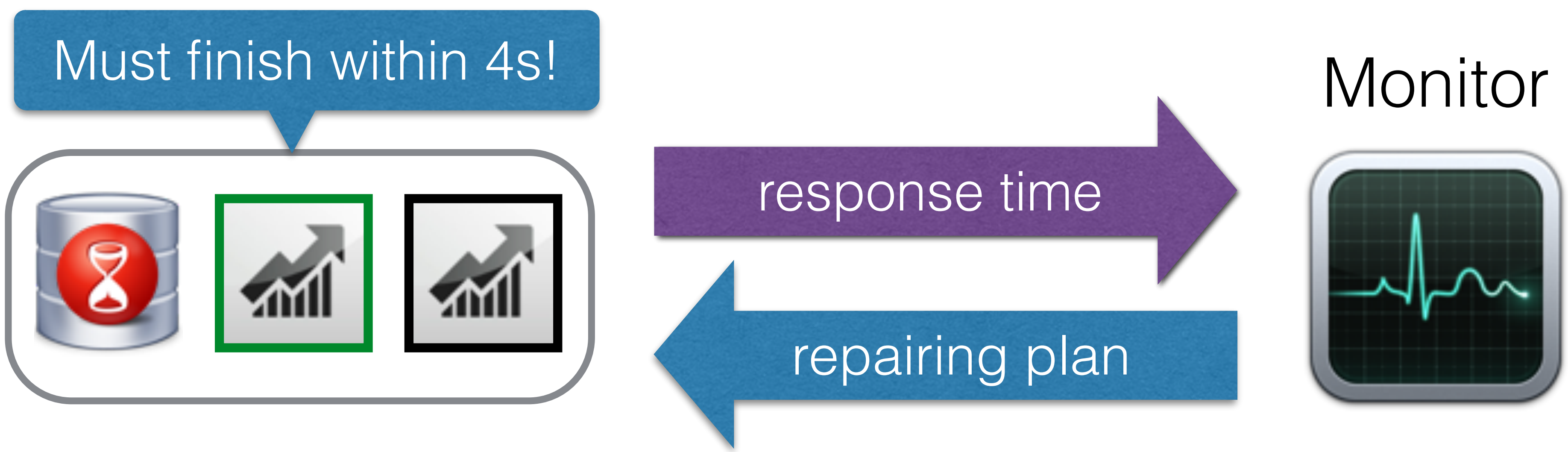
uLTR for runtime adaptation and recovery



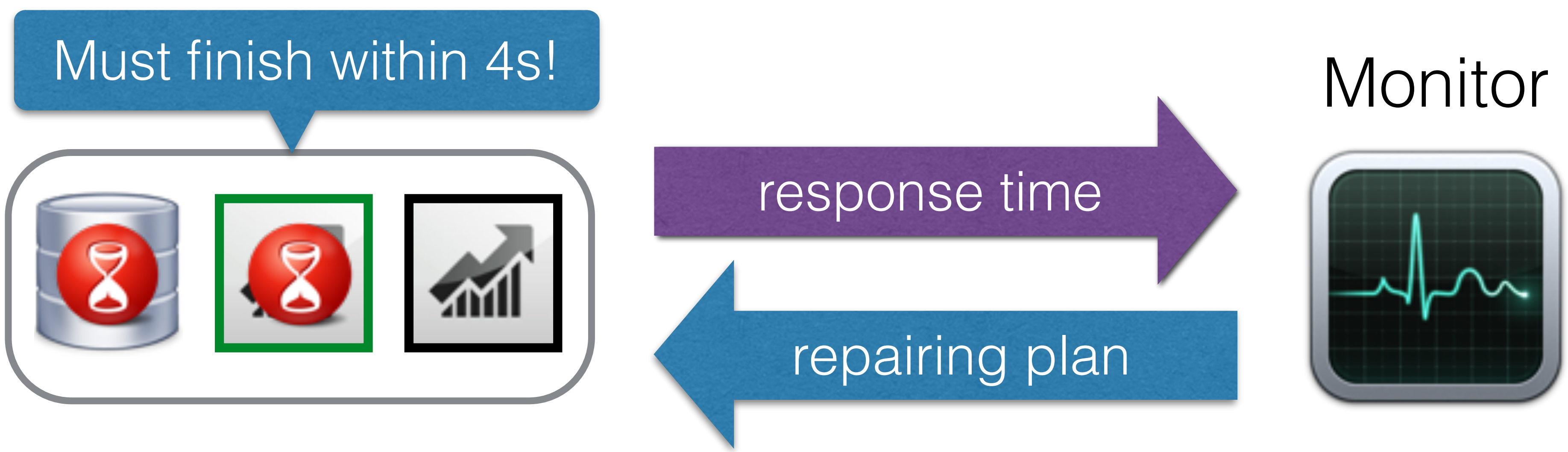
uLTR for runtime adaptation and recovery



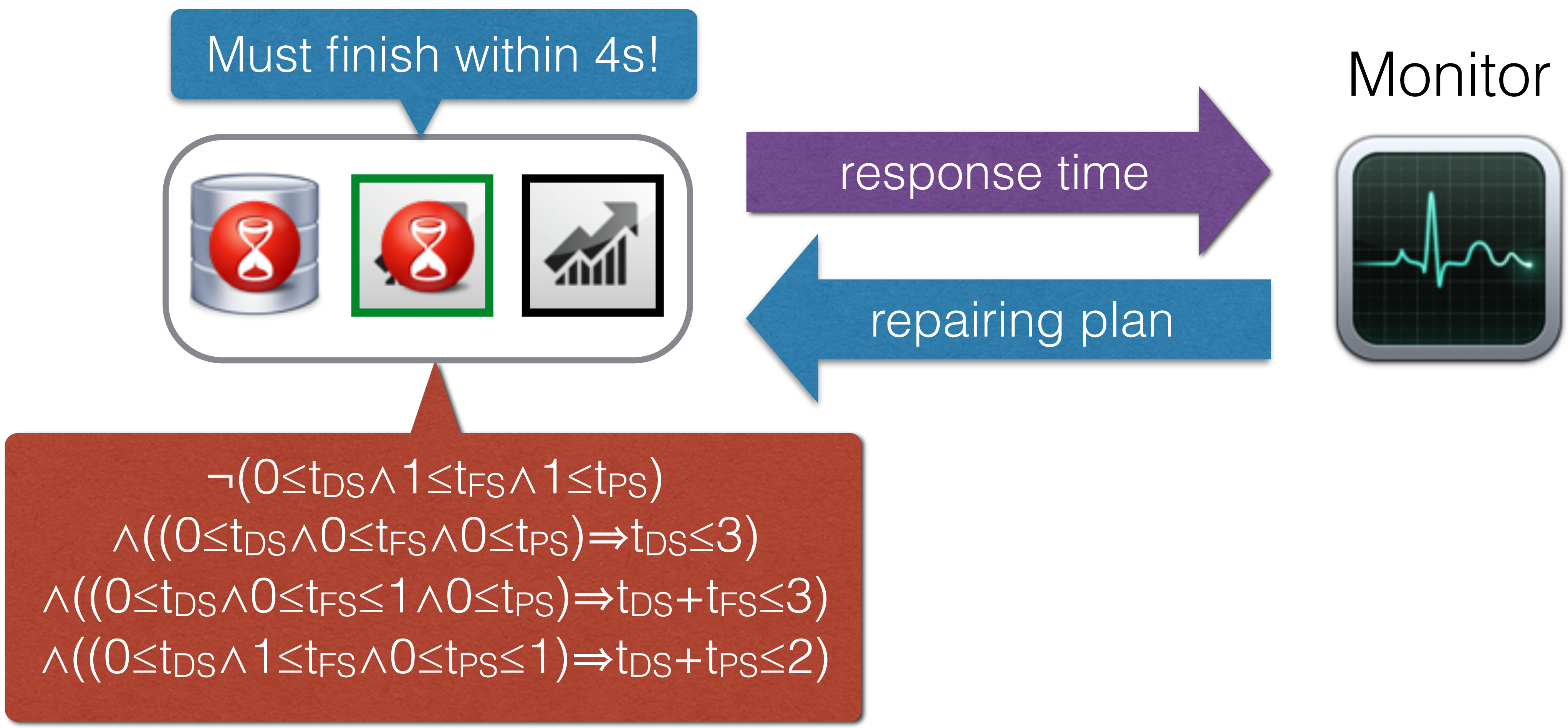
uLTR for runtime adaptation and recovery



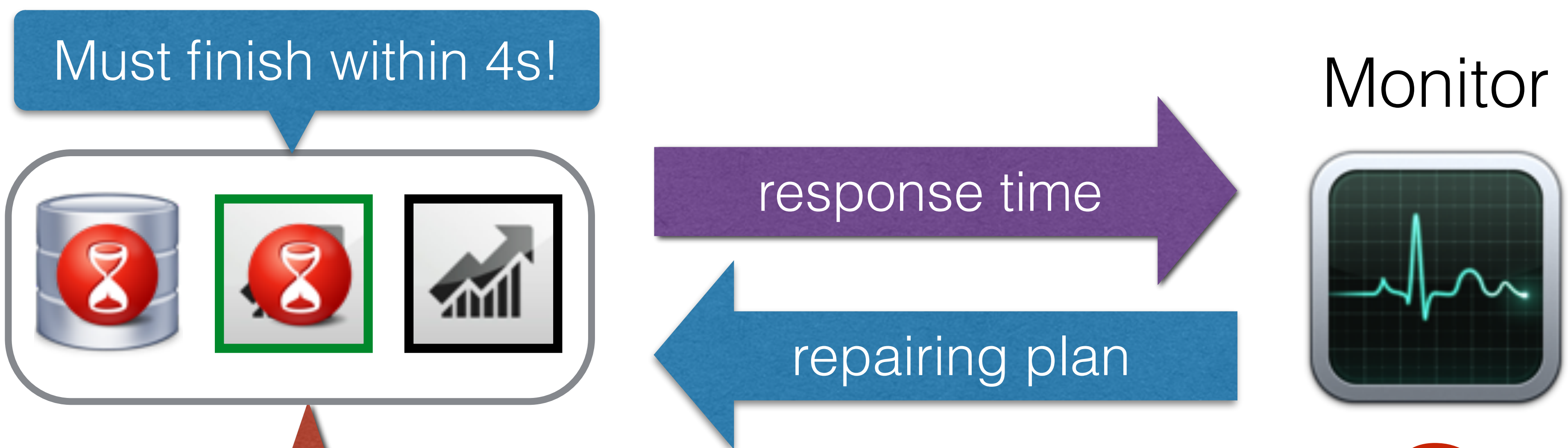
uLTR for runtime adaptation and recovery



uLTR for runtime adaptation and recovery



uLTR for runtime adaptation and recovery

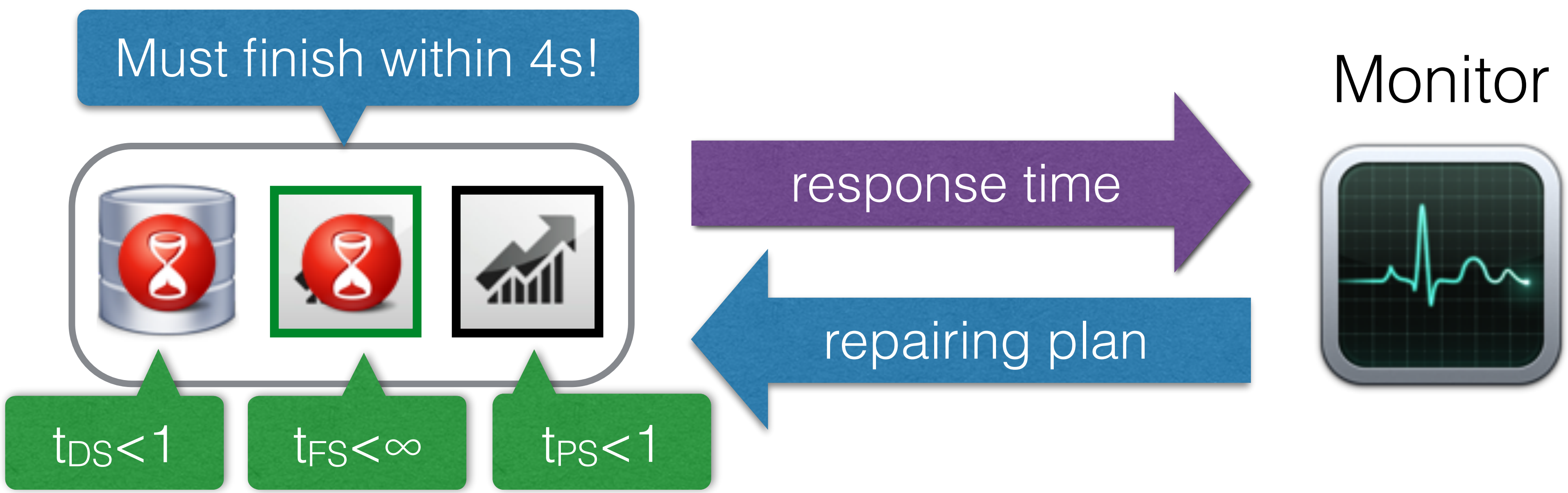


$\neg(0 \le t_{DS} \wedge 1 \le t_{FS} \wedge 1 \le t_{PS})$
$\wedge((0 \le t_{DS} \wedge 0 \le t_{FS} \wedge 0 \le t_{PS}) \Rightarrow t_{DS} \le 3)$
$\wedge((0 \le t_{DS} \wedge 0 \le t_{FS} \le 1 \wedge 0 \le t_{PS}) \Rightarrow t_{DS} + t_{FS} \le 3)$
$\wedge((0 \le t_{DS} \wedge 1 \le t_{FS} \wedge 0 \le t_{PS} \le 1) \Rightarrow t_{DS} + t_{PS} \le 2)$

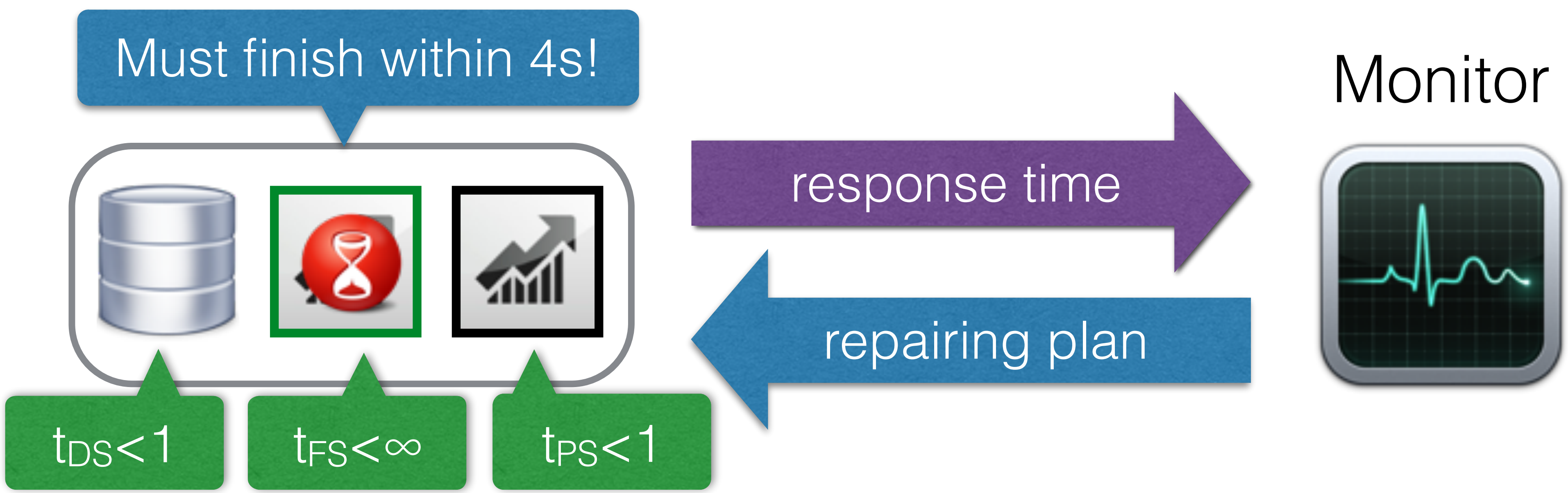
?

Have to replace both DS and FS.

uLTR for runtime adaptation and recovery

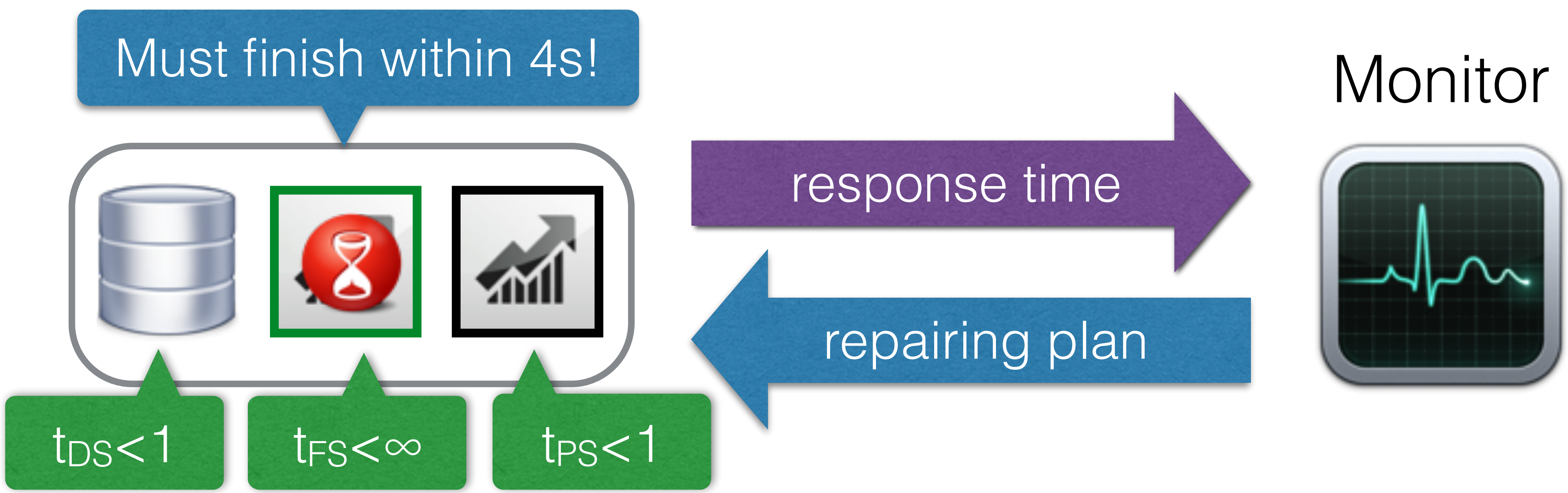


uLTR for runtime adaptation and recovery



Replacing DS is enough!

uLTR for runtime adaptation and recovery



Replacing DS is enough!

The "meaning" of LTR:
safe if one of t_{FS} and t_{PS} is less than 1.

uLTR for runtime adaptation and recovery

Experiments:

- *Use real service response time*
- *Simulate violations by adding uniform random delays to components*
- *Compare the length of recovery plans generated by LTR and uLTR*
- *In ~90% cases, uLTR discovers shorter repairs*

Limitations & Future Work

Limited evaluation

- *Need to look at other domains*

Proof of concept, not the silver bullet

- *Generalize the sampling algorithm: allow arbitrary hyper-rectangles*

Scalability issues:

- *Quantifier elimination*
- *Balance between precision and performance*

Checklist

☑ What is uLTR?

- *Component-independent under-approximated LTR*
- *Soundness: ensure timing safety*

☑ How to break up the monolithic constraint?

- *Compute uLTR from LTR*
- *Precision: preserve as many choices as possible*

☐ How can localized constraints support the management of timing requirements?

- *uLTR for component selection*
- *uLTR for runtime adaptation and recovery*

Checklist

What is uLTR?

- *Component-independent under-approximated LTR*
- *Soundness: ensure timing safety*

How to break up the monolithic constraint?

- *Compute uLTR from LTR*
- *Precision: preserve as many choices as possible*

How can localized constraints support the management of timing requirements?

- *uLTR for component selection*
- *uLTR for runtime adaptation and recovery*

Questions?

Thank you!

References

Li, Y., Albarghouthi, A., Gurfinkel, A., Kincaid, Z., Chechik, M.: Symbolic Optimization with SMT Solvers. In: Proc. of POPL 2014 (2014)

Tan, T.H., André, E., Sun, J., Liu, Y., Dong, J.S., Chen, M.: Dynamic Synthesis of Local Time Requirement for Service Composition. In: Proc. of ICSE 2013, pp. 542–551 (2013)

Al-Masri,E.,Mahmoud,Q.H.:QoS-based Discovery and Ranking of Web Services.In:Proc. of ICCCN 2007, pp. 529–534. IEEE (2007)

Wang, S., Rho, S., Mai, Z., Bettati, R., Zhao, W.: Real-time Component-based Systems. In: Proc. of RTETAS 2005, pp. 428–437 (2005)

Carminati, B., Ferrari, E., Hung, P.C.: Exploring Privacy Issues in Web Services Discovery Agencies. IEEE Security & Privacy 3(5), 14–21 (2005)