

# Verifying CSP-OZ-DC Specifications with Complex Data Types and Timing Parameters

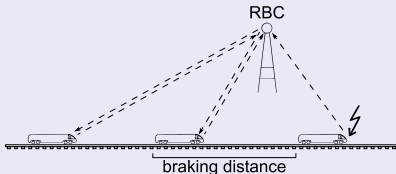
Integrated Formal Methods 2007, July 3rd

Johannes Faber<sup>1</sup>, Swen Jacobs<sup>2</sup>, and  
Viorica Sofronie-Stokkermans<sup>2</sup>

<sup>1</sup>University of Oldenburg  
j.faber@uni-oldenburg.de

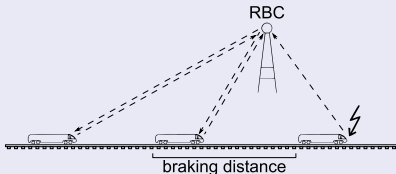
<sup>2</sup>Max-Planck-Institute Saarbrücken  
{sjacobs|sofronie}@mpi-inf.mpg.de

## Case Study: Radio Block Centre (RBC)



- RBC controls trains on track segment
- Number of trains kept as parameter
- Safety property: collision freedom

## Case Study: Radio Block Centre (RBC)



- RBC controls trains on track segment
- Number of trains kept as parameter
- Safety property: collision freedom

- Aim: High-level verification
- Specification: CSP-OZ-DC until now
  - without timing parameters
- Verification: Invariant checking, BMC until now
  - model checking, but only basic data types
  - unparametric number of components

## 1 Introduction

## 2 Specification

- CSP-OZ-DC
- Timing Parameters
- Semantics

## 3 Verification

- Hierarchic Reasoning
- Local Theory Extensions
- Application to Case Study

## 4 Conclusion

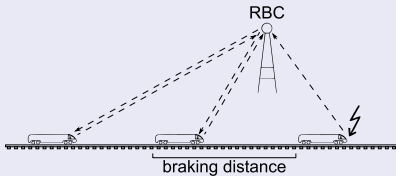
Introduction

Specification

Verification

Conclusion

## Case Study: Characteristics



Complex systems comprise

- processes running in parallel
- communications
- data space and data changes
- timing constraints

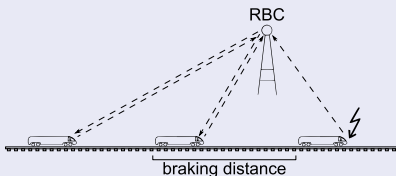
Introduction

**Specification**

Verification

Conclusion

## Case Study: Characteristics



Complex systems comprise

- processes running in parallel
- communications
- data space and data changes
- timing constraints

CSP-OZ-DC is a **combined** formalism that integrates

- Communicating Sequential Processes (**CSP**)
- Object-Z (**OZ**)
- Duration Calculus (**DC**)

to specify these aspects with the best-suited formalism

RBC[T\_PR : Q<sup>+</sup>]

```
method positionReport
method detectEm : [trainNumber : N]
  main ≜ Driveability ||| Detection
Driveability ≜ positionReport → Driveability
Detection ≜ detectEm → Detection
```

```
pos : seq Position          emTrain : N
speed : seq Speed          maxDec : Acceleration
maxSpeed, minSpeed : Speed d : Position
brakingDst : Speed → Position n : N

0 < minSpeed < maxSpeed
n = #pos = #speed
0 < d = brakingDst(maxSpeed)
∀ s : Speed • brakingDst(s) ≥  $\frac{s^2}{2 \cdot \text{maxDec}}$ 
∀ s1, s2 : Speed | s1 < s2 • brakingDst(s1) < brakingDst(s2)
brakingDst(0) = 0
```

```
Init
emTrain > n
∀ i : dom speed • minSpeed ≤ speed(i) ≤ maxSpeed
∀ i : dom pos | i ≠ 1
  • pos(i) < pos(i - 1) - brakingDst(speed(i))
```

```
¬(true ; ↑ positionReport ; (ℓ < T_PR) ; ↑ positionReport ; true)
¬(true ; ∃ positionReport ∧ (ℓ > T_PR) ; true)
```

com\_positionReport

```
Δ(pos, speed)
∀ i : dom pos | i = 1 ∧ i < emTrain
  • minSpeed ≤ speed'(i) ≤ maxSpeed
∀ i : dom pos | 1 < i < emTrain ∧ pos(i - 1) - pos(i) ≥ d
  • minSpeed ≤ speed'(i) ≤ maxSpeed
∀ i : dom pos | 1 < i < emTrain ∧ pos(i - 1) - pos(i) < d
  • minSpeed = speed'(i)
∀ i : dom pos | i ≥ emTrain
  • speed'(i) = max{speed(i) - maxDec * T_PR, 0}
∀ i : dom pos • pos'(i) = pos(i) + speed'(i) * T_PR
```

com\_detectEm

```
Δ(speed, emTrain)
newEmTrain? : N
newEmTrain? ≤ n
emTrain' = min{newEmTrain?, emTrain}
speed'(emTrain') = 0
∀ i : dom speed | i ≠ emTrain' • speed'(i) = speed(i)
```

} Interface

} CSP part

} OZ part

} DC part

Introduction

Specification

Verification

Conclusion

```

RBC[T_PR : Q+]
method positionReport
method detectEm : [trainNumber : N]
main ≡ Driveability ||| Detection
    
```

} Interface

## Interface and CSP Part

```

RBC[T_PR : Q+]
method positionReport
method detectEm : [trainNumber : N]
    main ≡ Driveability ||| Detection
    Driveability ≡ positionReport → Driveability
    Detection ≡ detectEm → Detection
    
```

```

¬(true ; ↑ positionReport ; (ℓ < T_PR) ; ↓ positionReport ; true)
¬(true ; ∃ positionReport ∧ (ℓ > T_PR) ; true)
    
```

} DC part

Introduction

Specification

Verification

Conclusion

```
RBC[T_PR : Q+]
method positionReport
method detectEm : [trainNumber : N]
main ⊆ Drivesability ||| Detection
```

} Interface

## State Schema

*pos* : seq *Position*  
*speed* : seq *Speed*  
*maxSpeed*, *minSpeed* : *Speed*  
*brakingDst* : *Speed* → *Position*

*emTrain* :  $\mathbb{N}$   
*maxDec* : *Acceleration*  
*d* : *Position*  
*n* :  $\mathbb{N}$

$$0 < \text{minSpeed} < \text{maxSpeed}$$

$$n = \#pos = \#speed$$

$$0 < d = \text{brakingDst}(\text{maxSpeed})$$

$$\forall s : \text{Speed} \bullet \text{brakingDst}(s) \geq \frac{s^2}{2 * \text{maxDec}}$$

$$\forall s_1, s_2 : \text{Speed} \mid s_1 < s_2 \bullet \text{brakingDst}(s_1) < \text{brakingDst}(s_2)$$

$$\text{brakingDst}(0) = 0$$

Introduction

Specification

Verification

Conclusion

```

RBC[T_PR : Q+]
method positionReport
method detectEm : [trainNumber : N]
  main ≙ Driveability ||| Detection
Driveability ≙ positionReport → Driveability
Detection ≙ detectEm → Detection
    
```

} Interface  
} CSP part

\_com\_positionReport\_

## Init Schema

Init

$emTrain > n$

$\forall i : \text{dom } speed \bullet minSpeed \leq speed(i) \leq maxSpeed$

$\forall i : \text{dom } pos \mid i \neq 1$

$\bullet pos(i) < pos(i - 1) - brakingDst(speed(i))$

$\neg(true ; \uparrow positionReport ; (\ell < T\_PR) ; \uparrow positionReport ; true)$   
 $\neg(true ; \exists positionReport \wedge (\ell > T\_PR) ; true)$

} DC part

```
RBC[T_PR : Q+]  
method positionReport  
method detectEm : [trainNumber : N]  
main 5. Drivability III Detection
```

} Interface

## Operation Schema *com\_positionReport*

*com\_positionReport*

$\Delta(pos, speed)$

$\forall i : \text{dom } pos \mid i = 1 \wedge i < emTrain$

- $\bullet \minSpeed \leq speed'(i) \leq maxSpeed$

$\forall i : \text{dom } pos \mid 1 < i < emTrain \wedge pos(i-1) - pos(i) \geq d$

- $\bullet \minSpeed \leq speed'(i) \leq maxSpeed$

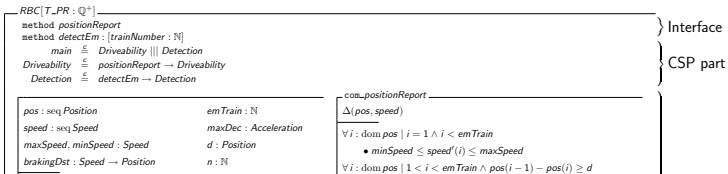
$\forall i : \text{dom } pos \mid 1 < i < emTrain \wedge pos(i-1) - pos(i) < d$

- $\bullet \minSpeed = speed'(i)$

$\forall i : \text{dom } pos \mid i \geq emTrain$

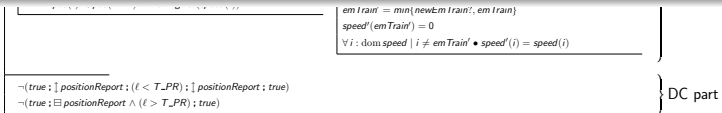
- $\bullet speed'(i) = \max\{speed(i) - maxDec * T\_PR, 0\}$

$\forall i : \text{dom } pos \bullet pos'(i) = pos(i) + speed'(i) * T\_PR$



## DC Part

$$\neg(\text{true}; \uparrow \text{positionReport}; (\ell < T\_PR); \uparrow \text{positionReport}; \text{true})$$

$$\neg(\text{true}; \exists \text{positionReport} \wedge (\ell > T\_PR); \text{true})$$


```

RBC[T_PR : Q+]
method positionReport
method detectEm : [trainNumber : N]
  main ≙ Driveability ||| Detection
Driveability ≙ positionReport → Driveability
Detection ≙ detectEm → Detection
  
```

} Interface  
} CSP part

## Operation Schema *com\_detectEm*

*com\_detectEm*

$\Delta(speed, emTrain)$

$newEmTrain? : \mathbb{N}$

$newEmTrain? \leq n$

$emTrain' = \min\{newEmTrain?, emTrain\}$

$speed'(emTrain') = 0$

$\forall i : \text{dom } speed \mid i \neq emTrain' \bullet speed'(i) = speed(i)$

$\neg(\text{true}; \exists positionReport \wedge (\ell > T\_PR); \text{true})$

} DC part

Introduction

Specification

Verification

Conclusion

## Case Study: Timing Parameters

$RBC[T\_PR : \mathbb{Q}^+]$

⋮

$com\_positionReport$

⋮

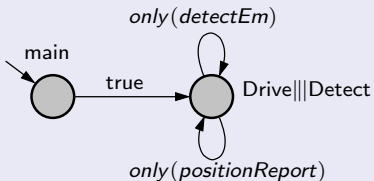
$\forall i : \text{dom } pos \bullet pos'(i) = pos(i) + speed'(i) * T\_PR$

⋮

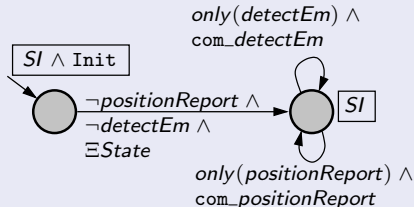
$\neg(true ; \uparrow positionReport ; (\ell < T\_PR) ; \uparrow positionReport ; true)$

$\neg(true ; \boxplus positionReport \wedge (\ell > T\_PR) ; true)$

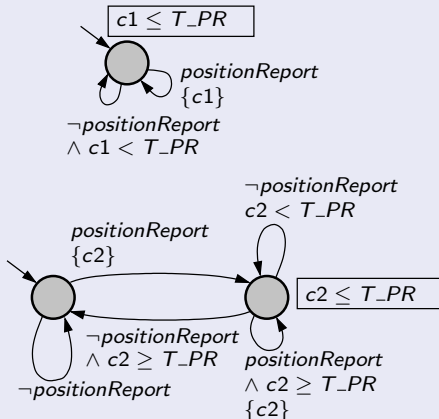
## Case Study: PEA for CSP Part



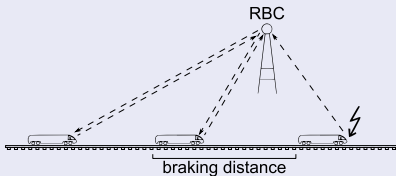
## Case Study: PEA for OZ Part



## Case Study: PEA for DC Part



## Case Study: Constraint Based Representation



$\text{com\_detectEm}$

$\Delta(\text{speed}, \text{emTrain})$

$\text{newEmTrain?} : \mathbb{N}$

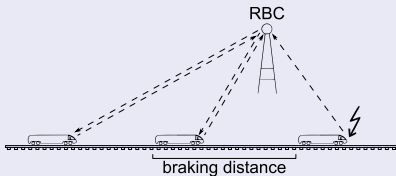
$\text{newEmTrain?} \leq n$

$\text{emTrain}' = \min\{\text{newEmTrain?}, \text{emTrain}\}$

$\text{speed}'(\text{emTrain}') = 0$

$\forall i \in \mathcal{I} \mid i \neq \text{emTrain}' \bullet \text{speed}'(i) = \text{speed}(i)$

## Case Study: Constraint Based Representation



$\text{com\_detectEm}$

$\Delta(\text{speed}, \text{emTrain})$

$\text{newEmTrain?} : \mathbb{N}$

$\text{newEmTrain?} \leq n$

$\text{emTrain}' = \min\{\text{newEmTrain?}, \text{emTrain}\}$

$\text{speed}'(\text{emTrain}') = 0$

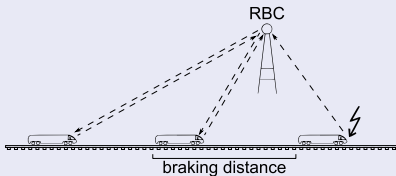
$\forall i \in \mathcal{I} \mid i \neq \text{emTrain}' \bullet \text{speed}'(i) = \text{speed}(i)$

Transition Constraints given by  $\Phi$

Train speeds are stored in an array  $\text{speed} : \mathcal{I} \rightarrow \mathbb{R}$

Update rules:  $\forall i \in \mathcal{I} : i \neq \text{emTrain}' \rightarrow \text{speed}'(i) = \text{speed}(i)$

## Case Study: Constraint Based Representation



$\text{com\_detectEm}$

$\Delta(\text{speed}, \text{emTrain})$

$\text{newEmTrain?} : \mathbb{N}$

$\text{newEmTrain?} \leq n$

$\text{emTrain}' = \min\{\text{newEmTrain?}, \text{emTrain}\}$

$\text{speed}'(\text{emTrain}') = 0$

$\forall i \in \mathcal{I} \mid i \neq \text{emTrain}' \bullet \text{speed}'(i) = \text{speed}(i)$

Transition Constraints given by  $\Phi$

Train speeds are stored in an array  $\text{speed} : \mathcal{I} \rightarrow \mathbb{R}$

Update rules:  $\forall i \in \mathcal{I} : i \neq \text{emTrain}' \rightarrow \text{speed}'(i) = \text{speed}(i)$

Safety as invariance property:

$\Psi = \forall i : 1 < i \leq n \rightarrow \text{pos}(i) < \text{pos}(i-1) - \text{brakingDst}(\text{speed}(i))$

**Task:** Check **unsatisfiability** of  $\mathbb{R} \cup \mathcal{I} \cup \Psi \cup \Phi \cup \neg\Psi'$

**Problem:** Standard methods do not support checking satisfiability of **universally quantified formulae over combinations of theories**

**Task:** Check **unsatisfiability** of  $\mathbb{R} \cup \mathcal{I} \cup \Psi \cup \Phi \cup \neg\Psi'$

**Problem:** Standard methods do not support checking satisfiability of **universally quantified formulae over combinations of theories**

- Idea:**
- replace quantified formulae with ground instances  
     $\rightsquigarrow$  **locality**
  - reduce task to a ground problem over  $\mathbb{R} \cup \mathcal{I}$
  - consider  $\Psi$  and  $\Phi$  as **local extensions** of the combined theory of  $\mathbb{R} \cup \mathcal{I}$   
     $\rightsquigarrow$  **decidable problem**

**Theory Extensions:** If  $\mathcal{T}_0$  is a theory,  $\mathcal{K}$  a set of clauses with new function symbols, then  $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$  is a **theory extension**.

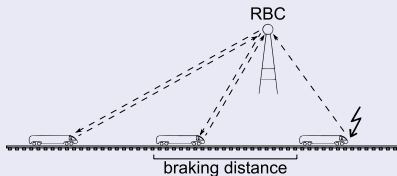
**Extension term:** A term that contains a new function symbol.

**Locality:** A theory extension  $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$  is **local** if for every ground goal  $G$

$$\mathcal{T}_0 \cup \mathcal{K} \cup G \models \perp \quad \Leftrightarrow \quad \mathcal{T}_0 \cup \mathcal{K}[G] \cup G \models \perp,$$

where  $\mathcal{K}[G]$  consists of the instances of axioms in  $\mathcal{K}$  with ground extension terms that appear in  $G$  or  $\mathcal{K}$ .

## Case Study: $\Phi_E$



*com\_detectEm*

$\Delta(\text{speed}, \text{emTrain})$

*newEmTrain?* :  $\mathbb{N}$

*newEmTrain?*  $\leq n$

*emTrain'* =  $\min\{\text{newEmTrain?}, \text{emTrain}\}$

*speed'*(*emTrain'*) = 0

$\forall i \in \mathcal{I} \mid i \neq \text{emTrain}' \bullet \text{speed}'(i) = \text{speed}(i)$

}  $\Phi_E$

$\mathbb{R} \cup \mathcal{I}$  is a theory,  $\Phi, \Phi_E$  are sets of clauses with new function symbols, so  $\mathbb{R} \cup \mathcal{I} \subseteq \mathbb{R} \cup \mathcal{I} \cup \Phi \cup \Phi_E$  is a **theory extension**.

An **extension term** is any term that contains *speed'*.

To simplify matters, we assume that  $\Phi$  does not contain any extension term.

## Case Study: $\Phi_E$

$$\begin{array}{l}
 \text{--- com\_detectEm ---} \\
 \Delta(\text{speed}, \text{emTrain}) \\
 \text{newEmTrain?} : \mathbb{N} \\
 \hline
 \left. \begin{array}{l}
 \text{newEmTrain?} \leq n \\
 \text{emTrain}' = \min\{\text{newEmTrain?}, \text{emTrain}\} \\
 \text{speed}'(\text{emTrain}') = 0 \\
 \forall i \in \mathcal{I} \mid i \neq \text{emTrain}' \bullet \text{speed}'(i) = \text{speed}(i)
 \end{array} \right\} \Phi_E
 \end{array}$$

$\mathbb{R} \cup \mathcal{I} \subseteq \mathbb{R} \cup \mathcal{I} \cup \Phi \cup \Phi_E$  is **local**, so for every ground goal  $G$

$$\mathbb{R} \cup \mathcal{I} \cup \Phi \cup \Phi_E \cup G \models \perp \quad \Leftrightarrow \quad \mathbb{R} \cup \mathcal{I} \cup \Phi \cup \Phi_E[G] \cup G \models \perp$$

Introduction

Specification

Verification

Conclusion

## Case Study: $\Phi_E$

<pre> com_detectEm Δ(speed, emTrain) newEmTrain? : ℕ </pre>	$\left. \begin{array}{l} \\ \\ \\ \end{array} \right\} \Phi_E$
<pre> newEmTrain? ≤ n emTrain' = min{newEmTrain?, emTrain} speed'(emTrain') = 0 </pre>	
<pre> ∀ i ∈ I   i ≠ emTrain' • speed'(i) = speed(i) </pre>	

$\mathbb{R} \cup \mathcal{I} \subseteq \mathbb{R} \cup \mathcal{I} \cup \Phi \cup \Phi_E$  is **local**, so for every ground goal  $G$

$\mathbb{R} \cup \mathcal{I} \cup \Phi \cup \Phi_E \cup G \models \perp \Leftrightarrow \mathbb{R} \cup \mathcal{I} \cup \Phi \cup \Phi_E[G] \cup G \models \perp$

We consider  $G = \neg\Psi' =$

$$\{1 < k_1, k_1 \leq n, k_2 = k_1 - 1, s = \text{speed}'(k_1), \\ \text{pos}'(k_1) \geq \text{pos}'(k_2) - \text{brakingDst}(s)\}$$

## Case Study: $\Phi_E$

```

com_detectEm
┌───────────
Δ(speed, emTrain)
newEmTrain? : ℕ
└──────────
newEmTrain? ≤ n
emTrain' = min{newEmTrain?, emTrain}
speed'(emTrain') = 0
∀ i ∈ I | i ≠ emTrain' • speed'(i) = speed(i)
} ΦE

```

$$\Phi_E[G] =$$

$$\begin{aligned}
& newEmTrain \leq n \wedge \\
& emTrain' = \min\{newEmTrain, emTrain\} \wedge \\
& speed'(emTrain') = 0 \wedge \\
& k_1 \neq emTrain' \rightarrow speed'(k_1) = speed(k_1) \wedge \\
& emTrain' \neq emTrain \\
& \rightarrow speed'(emTrain') = speed(emTrain')
\end{aligned}$$

$\mathbb{R} \cup \mathcal{I} \subseteq \mathbb{R} \cup \mathcal{I} \cup \Phi \cup \Phi_E$  is **local**, so for every ground goal  $G$

$$\mathbb{R} \cup \mathcal{I} \cup \Phi \cup \Phi_E \cup G \models \perp \quad \Leftrightarrow \quad \mathbb{R} \cup \mathcal{I} \cup \Phi \cup \Phi_E[G] \cup G \models \perp$$

We consider  $G = \neg\Psi' =$

$$\{1 < k_1, k_1 \leq n, k_2 = k_1 - 1, s = speed'(k_1),$$

$$pos'(k_1) \geq pos'(k_2) - brakingDst(s)\}$$



**Now:** Quantifiers removed from extension terms in  $\Phi_E$

**Next step:** Replace ground extension terms with constants

**Now:** Quantifiers removed from extension terms in  $\Phi_E$

**Next step:** Replace ground extension terms with constants

## Case Study: *speed'* and *pos'* removed

$$\mathit{speed}'(\mathit{emTrain}') \rightsquigarrow c_1$$

$$\mathit{speed}'(k_1) \rightsquigarrow c_2$$

$$G_0 = \{1 < k_1, k_1 \leq n, k_2 = k_1 + 1, s = c_2, \\ c_3 \geq c_4 - \mathit{brakingDst}(s)\}$$

$$\Phi_E[G]_0 = \mathit{newEmTrain} \leq n \wedge \\ \mathit{emTrain}' = \min\{\mathit{newEmTrain}, \mathit{emTrain}\} \wedge \\ c_1 = 0 \wedge k_1 \neq \mathit{emTrain}' \rightarrow c_2 = \mathit{speed}(k_1) \wedge \\ \mathit{emTrain}' \neq \mathit{emTrain}' \rightarrow c_1 = \mathit{speed}(\mathit{emTrain}')$$

$$N_0 = \{ \mathit{emTrain}' = k_1 \rightarrow c_1 = c_2, k_1 = k_2 \rightarrow c_3 = c_4 \}$$

**Now:** Quantifiers removed from extension terms in  $\Phi_E$

**Next step:** Replace ground extension terms with constants

## Case Study: *speed'* and *pos'* removed

$$speed'(emTrain') \rightsquigarrow c_1$$

$$speed'(k_1) \rightsquigarrow c_2$$

$$G_0 = \{1 < k_1, k_1 \leq n, k_2 = k_1 + 1, s = c_2, \\ c_3 \geq c_4 - brakingDst(s)\}$$

$$\Phi_E[G]_0 = newEmTrain \leq n \wedge \\ emTrain' = \min\{newEmTrain, emTrain\} \wedge \\ c_1 = 0 \wedge k_1 \neq emTrain' \rightarrow c_2 = speed(k_1) \wedge \\ emTrain' \neq emTrain' \rightarrow c_1 = speed(emTrain')$$

$$N_0 = \{emTrain' = k_1 \rightarrow c_1 = c_2, k_1 = k_2 \rightarrow c_3 = c_4\}$$

Introduction

Specification

Verification

Conclusion

**Now:** Quantifiers removed from extension terms in  $\Phi_E$

**Next step:** Replace ground extension terms with constants

## Case Study: *speed'* and *pos'* removed

$$speed'(emTrain') \rightsquigarrow c_1$$

$$speed'(k_1) \rightsquigarrow c_2$$

$$G_0 = \{1 < k_1, k_1 \leq n, k_2 = k_1 + 1, s = c_2, \\ c_3 \geq c_4 - brakingDst(s)\}$$

$$\Phi_E[G]_0 = newEmTrain \leq n \wedge \\ emTrain' = \min\{newEmTrain, emTrain\} \wedge \\ c_1 = 0 \wedge k_1 \neq emTrain' \rightarrow c_2 = speed(k_1) \wedge \\ emTrain' \neq emTrain' \rightarrow c_1 = speed(emTrain')$$

$$N_0 = \{emTrain' = k_1 \rightarrow c_1 = c_2, k_1 = k_2 \rightarrow c_3 = c_4\}$$

**Now:** Quantifiers removed from extension terms in  $\Phi_E$

**Next step:** Replace ground extension terms with constants

## Case Study: *speed'* and *pos'* removed

$$speed'(emTrain') \rightsquigarrow c_1 \qquad speed'(k_1) \rightsquigarrow c_2$$

$$G_0 = \{1 < k_1, k_1 \leq n, k_2 = k_1 + 1, s = c_2, \\ c_3 \geq c_4 - brakingDst(s)\}$$

$$\Phi_E[G]_0 = newEmTrain \leq n \wedge \\ emTrain' = \min\{newEmTrain, emTrain\} \wedge \\ c_1 = 0 \wedge k_1 \neq emTrain' \rightarrow c_2 = speed(k_1) \wedge \\ emTrain' \neq emTrain' \rightarrow c_1 = speed(emTrain')$$

$$N_0 = \{emTrain' = k_1 \rightarrow c_1 = c_2, k_1 = k_2 \rightarrow c_3 = c_4\}$$

**Result:** Verification task reduced to satisfiability problem  
over  $\mathbb{R} \cup \mathcal{I} \cup \Phi$  :  $\Phi_E[G]_0 \wedge G_0 \wedge N_0$

- Our case study can be represented by local theory extensions

$$\mathcal{T}_0 = \mathbb{R} \cup \mathcal{I}$$

$\mathcal{T}_1$  extends  $\mathcal{T}_0$  with function *brakingDst*

$\mathcal{T}_2$  extends  $\mathcal{T}_1$  with function *speed*

$\mathcal{T}_3$  extends  $\mathcal{T}_2$  with function *secure*

$\mathcal{T}_4$  extends  $\mathcal{T}_3$  with function *pos*

$\mathcal{T}_5$  extends  $\mathcal{T}_4$  with functions *pos'* and *speed'*

- For safety property  $\Psi$  hierarchical reasoning allows reduction to satisfiability problem over  $\mathcal{T}_0$ 
  - in applications, local theory extensions can often be found
  - so, efficient verification possible

- Our case study can be represented by local theory extensions

$$\mathcal{T}_0 = \mathbb{R} \cup \mathcal{I}$$

$\mathcal{T}_1$  extends  $\mathcal{T}_0$  with function *brakingDst*

$\mathcal{T}_2$  extends  $\mathcal{T}_1$  with function *speed*

$\mathcal{T}_3$  extends  $\mathcal{T}_2$  with function *secure*

$\mathcal{T}_4$  extends  $\mathcal{T}_3$  with function *pos*

$\mathcal{T}_5$  extends  $\mathcal{T}_4$  with functions *pos'* and *speed'*

- For safety property  $\Psi$  hierarchical reasoning allows reduction to satisfiability problem over  $\mathcal{T}_0$ 
  - in applications, local theory extensions can often be found
  - so, efficient verification possible
- Currently, reduction performed manually
- But **automatisation possible** (prototypical tool support)

- Our goal was to verify high-level specification with a parametric number of components
  - Extended CSP-OZ-DC with timing parameters to allow more flexible specifications
  - Used hierarchical reasoning to reduce these systems to a decidable base theory
    - ~> Verification with invariant checking and BMC

- Our goal was to verify high-level specification with a parametric number of components
  - Extended CSP-OZ-DC with timing parameters to allow more flexible specifications
  - Used hierarchical reasoning to reduce these systems to a decidable base theory
    - ↪ Verification with invariant checking and BMC
- Applied these method to a case study motivated by the European Train Control System
  - Also considered more complex examples, where time passes between position and speed updates

- Our goal was to verify high-level specification with a parametric number of components
  - Extended CSP-OZ-DC with timing parameters to allow more flexible specifications
  - Used hierarchical reasoning to reduce these systems to a decidable base theory
    - ↪ Verification with invariant checking and BMC
- Applied these method to a case study motivated by the European Train Control System
  - Also considered more complex examples, where time passes between position and speed updates
- Ongoing work
  - Tool support
  - Integration to abstraction refinement model checker