Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 13, no. 8, December 2017, pp. 1219-1232
DOI: 10.23940/ijpe.17.08.p5.12191232

ST-LUSTRE: A Novel Spatio-Temporal Language Towards Safety-Critical Cyber-Physical Systems

Jing Liu[a], Junyang Wang[a], Zhiwei Li[a], Haiying Sun[a,*],
Yuejun Wang[b], Dehui Du[a], Xiaohong Chen[a], Mingsong Chen[a]

[a]*National Trustworthy Embedded Software Engineering Reseearch Centre, East China Normal University, Shanghai 200333, China*
[b] *Shanghai Fuxin Intelligent Transportation Solutions Co.Ltd, Zhangjiang Hi-Tech Park, Shanghai 200120, China*

Abstract

Safety-Critical Cyber-Physical Systems (SCCPSs) are a special kind of Cyber-Physical Systems (CPSs) which highlight the importance of system correctness and safety. To apply automatic testing or model checking technique in CPSs, a model that fully captures the features is required to serve as input. So, a novel efficient spatio-temporal language and the analysis techniques are demanded to support both temporal and spatial expression and reasoning. In fact, a synchronous language, LUSTRE, is widely used in safety-critical systems development. However, LUSTRE lacks spatial constructors. Thus, it is difficult to express the behaviors related to spatial features in SCCPSs. In this paper, we propose a language named ST-LUSTRE to support the unified modeling of spatial and temporal properties of CPSs. We define the syntax and semantics of ST-LUSTRE. Its semantics is interpreted on the topological space and natural number which is based on time sets. We also specify typical SCCPSs properties in ST-LUSTRE. ST-LUSTRE is successfully applied to a communication based train control system of Shanghai Fuxin Intelligent Transportation Solutions CO.,Ltd. (FITSCO).

Keywords: lustre; cyber-physical system

(Submitted on October 11, 2017; Revised on November 9, 2017; Accepted on November 25, 2017)

1. Introduction

Cyber-Physical Systems (CPSs) are multidimensional and complex systems, which are integrations of computation and physical processes [9]. CPSs interact with the physical world and they must operate dependably, safely, securely, efficiently and in real-time [14]. Safety-critical Cyber-Physical Systems (SCCPSs) are a special kind of CPSs. Generally, SCCPSs involve close interactions between the two worlds and a certain change in one world should be reflected in the other world in a time-sensitive and/or spatial-sensitive manner [16].

Manual inspecting is high cost in this kind of system for the complex control logic, so it's necessary to describe the system in a formal language to achieve automatic testing or verification technique like model checking. LUSTRE is a synchronous data-flow language for programming systems which interact with their environments in real-time. Its main application field is to program automatic control and signal processing systems [5]. Synchrony means that a program reacts instantaneously to external events and the dataflow language means that a system is made up of a network of operators acting in parallel and in time with their inputs. In LUSTRE, all the entities can be interpreted as functions of time. A basic entity is a couple made up of (i) a sequence of values with a given type and (ii) a clock representing a suite of graduations [8]. As concurrency, simplicity and synchrony have been considered at the beginning of the development, the language is accepted by the industries.

RCC-8 presents eight jointly exhaustive and pairwise disjoint base relations between spatial regions, which is proposed by Egenhofer and Franzosa (1991) and Randell and his colleagues (1992) [13]. The RCC-8 calculus is intended for

* Corresponding author.
E-mail address: hysun@sei.ecnu.edu.cn

reasoning about spatial configurations. In RCC-8, regions are non-empty regular and closed subsets of a topological space, which can consist of more than one piece. The eight predicates are enough if we are interested in relationships between spatial regions. Furthermore, RCC-8 has nice computational properties. It turns out to be decidable [2].

Time and space are the basic features of our environment. SCCPSs integrate computation, communication and control with the physical world. Therefore, they are two essential ingredients of SCCPSs properties which need to be explicitly captured into models [15]. Furthermore, safety is crucial for safety-critical cyber-physical systems. Both spatial and temporal attributes should be considered to build more precise models. On the other hand, LUSTRE is quite suitable for programming SCCPSs. Due to its synchronicity, it can handle parallel problems well.

Based on these observations, we propose a formal spatio-temporal language named ST-LUSTRE to model safety-critical cyber-physical systems. ST-LUSTRE is an extension on LUSTRE language. Six operators are defined to model spatial relations: dc(disconnect), ec(externally connect), po(partially overlap), eq(identical with), tpp(tangential proper part), ntpp(nontangential proper part). The semantics of those mentioned above is interpreted on the topological space. We also define spatial variable τ in ST-LUSTRE and to satisfy original features of LUSTRE language, spatial variable τ is defined as a sequence of values.

The remains of the paper are organized as follows. Section 2 reviews spatial logic and the language LUSTRE. Section 3 presents the syntax of ST-LUSTRE; Section 4 presents the semantics of ST-LUSTRE. To show how to use ST-LUSTRE, in section 5, we express safety-critical properties by ST-LUSTRE. The real application and a tool are presented in Section 6. Section 7 presents related work. Finally, section 8 presents the conclusion and the future work.

2. Spatial logic and LUSTRE

In this section, we introduce spatial logic RCC-8 and the language LUSTRE as preliminaries.

2.1. Spatial logic RCC-8

RCC-8 is a first-order theory for qualitative spatial representation, which is designed by Randell, Cui and John [13]. The theory RCC-8 assumes one primitive dyadic relation: C(x,y), which is read as 'x connects with y'. The relation C(x,y) is reflexive and symmetric. Therefore, there exists two axioms: $(1)\forall xC(x,x)$; $(2)\forall xy[C(x,y) \rightarrow C(y,x)]$. Through using the primitive relation C(x,y), the other relations can be defined as follows:

$$DC(x,y) \equiv \neg C(x,y) \tag{1}$$

$$P(x,y) \equiv \forall z[C(z,x) \rightarrow C(z,y)] \tag{2}$$

$$PP(x,y) \equiv P(x,y) \wedge \neg P(y,x) \tag{3}$$

$$x = y \equiv P(x,y) \wedge P(y,x) \tag{4}$$

$$O(x,y) \equiv \exists z[P(z,x) \wedge P(y,z)] \tag{5}$$

$$PO(x,y) \equiv O(x,y) \wedge \neg P(x,y) \wedge \neg P(y,x) \tag{6}$$

$$DR(x,y) \equiv \neg O(x,y) \tag{7}$$

$$TPP(x,y) \equiv PP(x,y) \wedge \exists z[EC(z,x) \wedge EC(z,y)] \tag{8}$$

$$EC(x,y) \equiv C(x,y) \wedge \neg O(x,y) \tag{9}$$

$$NTPP(x,y) \equiv PP(x,y) \wedge \neg \exists z[EC(z,x) \wedge EC(z,y)] \tag{10}$$

$$P(-1)(x,y) \equiv P(y,x) \tag{11}$$

$$PP(-1)(x,y) \equiv PP(y,x) \tag{12}$$

$$TPP(-1)(x,y) \equiv TPP(y,x) \tag{13}$$

$$NTPP(-1)(x,y) \equiv NTPP(y,x) \tag{14}$$

where DC(x,y) means x and y are disconnected; P(x,y) means x is a part of y; PP(x,y) means x is a proper part of y; x=y means x is identical with y; O(x,y) means x overlaps y; PO(x,y) means x partially overlaps y; DR(x,y) means x is discrete from y; TPP(x,y) means x is a tangential proper part of y; EC(x,y) means x is externally connected with y; NTPP(x,y) means x is a nontangential proper part of y. And RCC-8 contains eight relations of these: DC(x,y), EC(x,y), PO(x,y), EQ(x,y), TPP(x,y), NTPP(x,y) and the inverse of TPP(x,y), NTPP(x,y).

2.2. The language LUSTRE

LUSTRE is a data flow synchronous language. LUSTRE has two primary features: one is that it is based on dataflow, and the other one is its synchronization. The dataflow characteristic enables it to be descripted using block diagrams. The synchronization makes it suitable for handling time in programs. All variables in LUSTRE are a function of time. For example, "x=y+z" means that at each instant t, $x_t = y_t + z_t$. All the values of a stream are the same type, which is called the type of the stream. In LUSTRE, any program or piece of program has a cyclic behavior. That cycle defines a sequence of times, which is called the basic clock of the program: a flow whose clock is the basic clock takes its nth value at the nth execution cycle of the program [8]. There are four kinds of usual operators in LUSTRE: boolean, arithmetic, comparison and conditional. All the operators operate pointwise on flows. The syntax and the semantics of LUSTRE are subset of those in ST-LUSTRE and thus we do not present them here, which will be presented in section 3 and section 4.

3. Syntax of ST-LUSTRE

In this section, we define the syntax of ST-LUSTRE. Based on Region Connection Calculus theory, we define a new variable type *Spatial* and six operators on the new variable type. The complete syntax of ST-LUSTRE is as follows:

$$e ::= e_a \mid e_s \mid pre(e) \mid e_a \rightarrow e_s \mid e_a when\ e_s \mid e_a current\ e_s \tag{15}$$

$$e_a ::= e_a \mid uop\ e_{a1} \mid e_{a1} bop\ e_{a2} \mid if\ e_{a1}\ then\ e_{a2}\ else\ e_{a3} \tag{16}$$

$$e_s ::= dc(\tau_1,\tau_2) \mid ec(\tau_1,\tau_2) \mid eq(\tau_1,\tau_2) \mid po(\tau_1,\tau_2) \mid tpp(\tau_1,\tau_2) \mid ntpp(\tau_1,\tau_2) \tag{17}$$

where ea is original expression in LUSTRE and es is spatial expression which is newly defined in ST-LUSTRE. uop is unary operator and bop is binary operator. Unary operator only contains not and binary operator contains: +, -, *, /, >, >=, <, <=, =, <>, and, or.

Operator "pre", "→", "when", "current" are temporal operators and their usage and meanings are as follows and we summarize them in Fig.1.

- The operator "pre" means getting the value of previous cycle. For example, if E = (e_1, e_2, e_3, ..., e_n, ...), then pre(E) = (nil, e_1, e_2, e_3, ..., e_{n-1}, ...), where nil is the undefined value.
- The operator "→" means initializing streams. For example, if A=(a_1, a_2, a_3, ... , a_n, ...) and E=(e_1, e_2, e_3, ... , e_n, ...), which are two streams of the same type, then "A→E" is the stream (a_1, e_2, e_3, ..., e_n ...).

A	a1	a2	a3	a4	a5	a6
E	e1	e2	e3	e4	e5	e6
pre(E)	nil	e1	e2	e3	e4	e5
A -> E	a1	e2	e3	e4	e5	e6

A	true	false	true	false	false	true
E	e1	e2	e3	e4	e5	e6
E when A	e1		e3			e6
E current A	e1	e1	e3	e3	e3	e6

Figure 1. Temporal operators in ST-LUSTRE

- If E is an expression and A is a boolean expression, the expression "E when A" means that it is the sequence of values of E when A is true. For example, if E=(e_1, e_2, e_3, ..., e_n, ...) and A=(true, false, true, ... , true ...), then (E when A) = (e_1, e_2, e_3, ..., e_n, ...)
- If E is an expression and A is a boolean expression, the expression "E current A" means that getting the value of E at the last time A was true. For example, if E=(e_1, e_2, e_3, ..., e_n, ...) and A=(true, false, true, ..., true, ...),then (E current A) = (e_1, e_1, e_3, ..., e_n, ...)

τ is a spatial variable of spatial type, which is newly defined in ST-LUSTRE. The value of spatial operators is also a sequence in time. For example, if $\tau 1 = (s_1, s_2, s_1, ...)$, $\tau 2 = (s_2, s_1, s_3, ...)$ and supposing that s_1 and s_2 partially overlap; s_1 and s_3 are disconnected, then $dc(\tau_1, \tau_2)$ = (f, f, t, ...), which is a sequence in bool type. The syntax of six spatial operators is as follows and we summarize them in Fig.2.

$$dc(\tau_1, \tau_2) = \neg\boxed{\exists}(\tau_1 \sqcap \tau_2) \tag{18}$$

$$ec(\tau_1, \tau_2) = \boxed{\exists}(\tau_1 \sqcap \tau_2) \wedge \neg\boxed{\exists}(\mathrm{I}\tau_1 \sqcap \mathrm{I}\tau_2) \tag{19}$$

$$po(\tau_1, \tau_2) = \boxed{\exists}(\mathrm{I}\tau_1 \sqcap \mathrm{I}\tau_2) \wedge \neg\boxed{\forall}(\tau_1 \sqsubset \tau_2) \wedge \neg\boxed{\forall}(\tau_2 \sqsubset \tau_1) \tag{20}$$

$$eq(\tau_1, \tau_2) = \boxed{\forall}(\tau_1 \sqsubset \tau_2) \wedge \boxed{\forall}(\tau_2 \sqsubset \tau_1) \tag{21}$$

$$tpp(\tau_1, \tau_2) = \boxed{\forall}(\tau_1 \sqsubset \tau_2) \wedge \neg\boxed{\forall}(\tau_2 \sqsubset \tau_1) \wedge \neg\boxed{\forall}(\tau_1 \sqsubset \mathrm{I}\tau_2) \tag{22}$$

$$ntpp(\tau_1, \tau_2) = \boxed{\forall}(\tau_1 \sqsubset \mathrm{I}\tau_2) \wedge \neg\boxed{\forall}(\tau_2 \sqsubset \tau_1) \tag{23}$$

where $\boxed{\exists}$, $\boxed{\forall}$ and I are operators in spatial logic S4u. $\boxed{\forall}\tau$ means τ occupies the whole space. $\boxed{\exists}\tau$ means there is at least one point in τ. I is the interior operator. $\sqcap$ means intersection. $\tau 1 \sqsubset \tau 2$ is an abbreviation for $\overline{\tau_1 \sqcap \tau_2}$.

If temporal operators are blended with spatial operators, for example, pre($dc(\tau_1, \tau_2)$), it means getting the value of $dc(\tau_1, \tau_2)$ in previous cycle. For example, if $dc(\tau_1, \tau_2)$ =(t, f, t, ...), then pre($dc(\tau_1, \tau_2)$) = (nil, t, f, t, ...).

4. Semantics of ST-LUSTRE

In this section, we define the semantics of ST-LUSTRE. The semantics of ST-LUSTRE can be interpreted by a topological temporal model. The topological temporal model is as follows:

$$\mathfrak{M} =< \mathfrak{T}, \mathcal{I}, \mathfrak{U} > \tag{24}$$

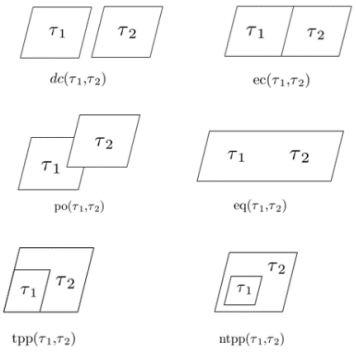

Figure 2. Six spatial operators in ST-LUSTRE

where $\mathfrak{T}$ is a flow of discrete time, $\mathcal{I}$ is a topological space and $\mathfrak{U}$ is a function, which can associate with every spatial term τ and every discrete time point w∈W onto a set $\mathfrak{U}$ (τ, w)⊆ U - the space occupied by τ at moment w; it can also associate with every common variable m onto a set $\mathfrak{U}$ (m)⊆U.

$$\mathfrak{U}(X) \triangleq \begin{cases} \mathfrak{U}(s,w) \subseteq U, X = (s,w) \\ \mathfrak{U}(p) \subseteq U, X = p \end{cases} \tag{25}$$

then we can achieve properties as follows:

$$\mathfrak{U}(\overline{\tau}, w) = U - \mathfrak{U}(\tau, w) \tag{26}$$

$$\mathfrak{U}(I\tau, w) = \mathbb{I}\mathfrak{U}(\tau, w), \tag{27}$$

$$\mathfrak{U}(\tau_1 \sqcap \tau_2, w) = \mathfrak{U}(\tau_1, w) \cap \mathfrak{U}(\tau_2, w) \tag{28}$$

where $\mathbb{I}$ is the interior operator and it is satisfied with the Kuratowski axioms.

Firstly, we define the semantics of spatial operators in ST-LUSTRE. These operators are newly defined in ST-LUSTRE and their semantics is as follows:

$$(\mathfrak{M}, w) \vDash dc(\tau_1, \tau_2) \; iff \; \forall w \in W, \neg\exists p \; p \in \tau_1 \cap \tau_2 \tag{29}$$

$$(\mathfrak{M}, w) \vDash ec(\tau_1, \tau_2) \; iff \; \forall w \in W, \exists p \; p \in \tau_1 \cap \tau_2 \wedge \neg\exists p \; p \in \mathbb{I}\tau_1 \cap \mathbb{I}\tau_2 \tag{30}$$

$$(\mathfrak{M}, w) \vDash eq(\tau_1, \tau_2) \; iff \; \forall w \in W, \forall (p \in \tau_1 \leftrightarrow p \in \tau_2) \tag{31}$$

$$(\mathfrak{M}, w) \vDash po(\tau_1, \tau_2) \; iff \; \forall w \in W, \exists p \; p \in \mathbb{I}\tau_1 \cap \mathbb{I}\tau_2 \wedge \exists p \; p \in \mathbb{I}\tau_1 \cap \neg\tau_2 \wedge \exists p \; p \in \neg\tau_1 \cap \mathbb{I}\tau_2 \tag{32}$$

$$(\mathfrak{M}, w) \vDash tpp(\tau_1, \tau_2) \; iff \; \forall w \in W, \forall p \; p \in \neg\tau_1 \cup \tau_2 \wedge \exists p \; p \in \tau_1 \cap \mathbb{I}\neg\tau_2 \wedge \exists p \; p \in \neg\tau_1 \cap \tau_2 \tag{33}$$

$$(\mathfrak{M}, w) \vDash ntpp(\tau_1, \tau_2) \; iff \; \forall w \in W, \forall p \; p \in \neg\tau_1 \cup \mathbb{I}\tau_2 \wedge \exists p \; p \in \neg\tau_1 \cap \tau_2 \tag{34}$$

where $\mathbb{I}$ is the interior operator.

Secondly, we define the semantics of temporal operators in ST-LUSTRE. They include four operators. Since the biggest difference between ST-LUSTRE and LUSTRE is spatial expression, we will only define the semantics of temporal operators operating on spatial expression. The semantics of temporal operators not operating on spatial expression in ST-LUSTRE is the same as those in LUSTRE, therefore we will not list them in this section.

pre:

$$(\mathfrak{M}, w) \vDash pre(dc(\tau_1, \tau_2)) \; iff \; (\mathfrak{M}, w-1) \vDash dc(\tau_1, \tau_2) \tag{35}$$

$$(\mathfrak{M}, w) \vDash pre(ec(\tau_1, \tau_2)) \; iff \; (\mathfrak{M}, w-1) \vDash ec(\tau_1, \tau_2) \tag{36}$$

$$(\mathfrak{M}, w) \vDash pre(po(\tau_1, \tau_2)) \; iff \; (\mathfrak{M}, w-1) \vDash po(\tau_1, \tau_2) \tag{37}$$

$$(\mathfrak{M}, w) \vDash pre(eq(\tau_1, \tau_2)) \; iff \; (\mathfrak{M}, w-1) \vDash eq(\tau_1, \tau_2) \tag{38}$$

$$(\mathfrak{M}, w) \vDash pre(tpp(\tau_1, \tau_2)) \; iff \; (\mathfrak{M}, w-1) \vDash tpp(\tau_1, \tau_2) \tag{39}$$

$$(\mathfrak{M}, w) \vDash pre(ntpp(\tau_1, \tau_2)) \; iff \; (\mathfrak{M}, w-1) \vDash ntpp(\tau_1, \tau_2) \tag{40}$$

These formulae above mean that if $\mathfrak{M}$ satisfies pre(e_1) at w instance, then $\mathfrak{M}$ satisfies e_1 at w-1 instance, where e_1 is a spatial formula.

followed by(→):

$$(\mathfrak{M}, w) \vDash e_1 \rightarrow dc(\tau_1, \tau_2) \; iff \; w \in W, w = 1, (\mathfrak{M}, w) \vDash e_1 \wedge \forall w > 1, (\mathfrak{M}, w) \vDash dc(\tau_1, \tau_2) \tag{41}$$

$$(\mathfrak{M}, w) \vDash e_1 \rightarrow ec(\tau_1, \tau_2) \; iff \; w \in W, w = 1, (\mathfrak{M}, w) \vDash e_1 \wedge \forall w > 1, (\mathfrak{M}, w) \vDash ec(\tau_1, \tau_2) \tag{42}$$

$$(\mathfrak{M}, w) \vDash e_1 \rightarrow eq(\tau_1, \tau_2) \; iff \; w \in W, w = 1, (\mathfrak{M}, w) \vDash e_1, \wedge \forall w > 1, (\mathfrak{M}, w) \vDash eq(\tau_1, \tau_2) \tag{43}$$

$$(\mathfrak{M}, w) \vDash e_1 \rightarrow po(\tau_1, \tau_2) \; iff \; w \in W, w = 1, (\mathfrak{M}, w) \vDash e_1 \wedge \forall w > 1, (\mathfrak{M}, w) \vDash po(\tau_1, \tau_2) \tag{44}$$

$$(\mathfrak{M}, w) \vDash e_1 \rightarrow tpp(\tau_1, \tau_2) \; iff \; w \in W, w = 1, (\mathfrak{M}, w) \vDash e_1 \wedge \forall w > 1, (\mathfrak{M}, w) \vDash tpp(\tau_1, \tau_2) \tag{45}$$

$$(\mathfrak{M}, w) \vDash e_1 \rightarrow ntpp(\tau_1, \tau_2) \; iff \; w \in W, w = 1, (\mathfrak{M}, w) \vDash e_1 \wedge \forall w > 1, (\mathfrak{M}, w) \vDash ntpp(\tau_1, \tau_2) \tag{46}$$

These formulae above mean that if $\mathfrak{M}$ satisfies $e_1 \rightarrow e_2$ at w instance, then $\mathfrak{M}$ satisfies e_1 at the first instance and at later instances, $\mathfrak{M}$ satisfies e_2, where e_2 is spatial formula. Besides, to avoid chaos, if the operator "followed by" operating on spatial expression, the left side of the operator must be bool type and can't be spatial expression and the spatial expression must be in the right side.

when:

$$\begin{aligned}&(\mathfrak{M}, w) \vDash e_1 when \; dc(\tau_1, \tau_2) \; iff \; \forall w \in W, (\mathfrak{M}, w) \vDash \\ &dc(\tau_1, \tau_2) \wedge (\mathfrak{M}, w) \vDash e_1 \vee \forall w \in W, \mathfrak{M} \nvDash dc(\tau_1, \tau_2) \wedge \mathfrak{M} \vDash nil\end{aligned} \tag{47}$$

$$(\mathfrak{M},w)\models e_1 when\ ec(\tau_1,\tau_2)\ iff\ \forall w\in W,(\mathfrak{M},w)\models ec(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_1\vee\forall w\in W,(\mathfrak{M},w)\not\models ec(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models nil \tag{48}$$

$$(\mathfrak{M},w)\models e_1 when\ eq(\tau_1,\tau_2)\ iff\ \forall w\in W,(\mathfrak{M},w)\models eq(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_1\vee\forall w\in W,(\mathfrak{M},w)\not\models eq(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models nil \tag{49}$$

$$(\mathfrak{M},w)\models e_1 when\ po(\tau_1,\tau_2)\ iff\ \forall w\in W,(\mathfrak{M},w)\models po(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_1\vee\forall w\in W,(\mathfrak{M},w)\not\models po(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models nil \tag{50}$$

$$(\mathfrak{M},w)\models e_1 when\ tpp(\tau_1,\tau_2)\ iff\ \forall w\in W,(\mathfrak{M},w)\models tpp(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_1\vee\forall w\in W,(\mathfrak{M},w)\not\models tpp(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models nil \tag{51}$$

$$(\mathfrak{M},w)\models e_1 when\ ntpp(\tau_1,\tau_2)\ iff\ \forall w\in W,(\mathfrak{M},w)\models ntpp(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_1\vee\forall w\in W,(\mathfrak{M},w)\not\models ntpp(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models nil \tag{52}$$

These formulae above mean that if $\mathfrak{M}$ satisfies e_1 when e_2 at w instance, then at w instance, either $\mathfrak{M}$ satisfies e_2 and $\mathfrak{M}$ satisfies e_1 or $\mathfrak{M}$ does not satisfy e_2 and $\mathfrak{M}$ satisfies nil, where e_2 is spatial formula. Besides, if the operator "when" operating on spatial expression, the spatial expression must be in the right side. The result will be the e_1 value of the cycle in which the spatial expression returns true.

current：

$$(\mathfrak{M},w)\models e_1\ current\ dc(\tau_1,\tau_2)\ iff\ (\forall w\in W,(\mathfrak{M},w)\models dc(\tau_1,\tau_2)\wedge(\mathfrak{M},w+1)\models dc(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_1)\vee(\forall w\in W,(\mathfrak{M},w)\models dc(\tau_1,\tau_2)\wedge(\mathfrak{M},w+1)\not\models dc(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_w) \tag{53}$$

$$(\mathfrak{M},w)\models e_1\ current\ ec(\tau_1,\tau_2)\ iff\ (\forall w\in W,(\mathfrak{M},w)\models ec(\tau_1,\tau_2)\wedge(\mathfrak{M},w+1)\models ec(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_1)\vee(\forall w\in W,(\mathfrak{M},w)\models ec(\tau_1,\tau_2)\wedge(\mathfrak{M},w+1)\not\models ec(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_w) \tag{54}$$

$$(\mathfrak{M},w)\models e_1\ current\ eq(\tau_1,\tau_2)\ iff\ (\forall w\in W,(\mathfrak{M},w)\models eq(\tau_1,\tau_2)\wedge(\mathfrak{M},w+1)\models eq(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_1\vee(\forall w\in W,(\mathfrak{M},w)\models eq(\tau_1,\tau_2)\wedge(\mathfrak{M},w+1)\not\models eq(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_w) \tag{55}$$

$$(\mathfrak{M},w)\models e_1\ current\ po(\tau_1,\tau_2)\ iff\ (\forall w\in W,(\mathfrak{M},w)\models po(\tau_1,\tau_2)\wedge(\mathfrak{M},w+1)\models po(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_1)\vee(\forall w\in W,(\mathfrak{M},w)\models po(\tau_1,\tau_2)\wedge(\mathfrak{M},w+1)\not\models po(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_w) \tag{56}$$

$$(\mathfrak{M},w)\models e_1\ current\ tpp(\tau_1,\tau_2)\ iff\ (\forall w\in W,(\mathfrak{M},w)\models tpp(\tau_1,\tau_2)\wedge(\mathfrak{M},w+1)\models tpp(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_1)\vee(\forall w\in W,(\mathfrak{M},w)\models tpp(\tau_1,\tau_2)\wedge(\mathfrak{M},w+1)\not\models tpp(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_w) \tag{57}$$

$$(\mathfrak{M},w)\models e_1\ current\ ntpp(\tau_1,\tau_2)\ iff\ (\forall w\in W,(\mathfrak{M},w)\models ntpp(\tau_1,\tau_2)\wedge(\mathfrak{M},w+1)\models ntpp(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_1)\vee(\forall w\in W,(\mathfrak{M},w)\models ntpp(\tau_1,\tau_2)\wedge(\mathfrak{M},w+1)\not\models tpp(\tau_1,\tau_2)\wedge(\mathfrak{M},w)\models e_w) \tag{58}$$

These formulae above mean that if $\mathfrak{M}$ satisfies e_1 current e_2 at w instance, then either $\mathfrak{M}$ satisfies e_2 at w instance and $\mathfrak{M}$ satisfies e_2 at w+1 instance and $\mathfrak{M}$ satisfies e_1 at w instance or $\mathfrak{M}$ satisfies e_2 at w instance and $\mathfrak{M}$ does not satisfy e_2 at w+1 instance and $\mathfrak{M}$ satisfies the value at w instance. Besides, the spatial expression must be on the right side of the operator "current" and the left side cannot be spatial expression. The "current" expression will return e_1 value at the last time the spatial expression returns true.

5. Specifying safety-critical properties with ST-LUSTRE

In SCCPSs, there are some safety-critical properties. We will use ST-LUSTRE expressing these properties in this part.

5.1. Safety properties

Safety property means that a formula should be met by all states in system. In ST-LUSTRE, all entities are the function of time; thus, safety can be described as the following form:

$$\alpha = true \tag{59}$$

where α is a ST-LUSTRE formula. This expression means in all cycles, formula α is always met. An example of safety property for spatial variable is the formula

$$po(\tau_1, \tau_2) = true \tag{60}$$

which means τ_1 and τ_2 always partially overlap in all cycles. An example of safety property for normal variable is the formula

$$(x \geq 0) = true \tag{61}$$

which means x is nonnegative in all states of the system.

5.2. Guarantee properties

Guarantee property means that a formula should be met by at least one state in the system. We define guarantee property as following form

$$not\ (\alpha = false) \tag{62}$$

This formula claims that at least one state satisfies α . An example of guarantee property for spatial variable is the following formula

$$not(dc(\tau_1, \tau_2) = false) \tag{63}$$

which means that τ_1 and τ_2 disconnect at least one state in the system. An example of guarantee property for normal variable is the following formula

$$not\ ((x \geq 0) = false) \tag{64}$$

which means x is nonnegative at least one state of the system.

5.3. Obligation properties

In computing process of the system, some properties can't be defined only by safety property or guarantee property; they must be defined by a logic combination of safety property or guarantee property, which is obligation property. We define obligation property as following form

$$(\alpha_1 = true)\ or\ (not(\alpha_2 = false)) \tag{65}$$

where α is a ST-LUSTRE formula. This formula means that either α_1 holds at all states of a computation or α_2 holds at some state. An example of obligation property for spatial variable is the following formula

$$(dc(\tau_1, \tau_2) = true)\ or\ (not(po(\tau_1, \tau_2) = false)) \tag{66}$$

which means that either τ_1 and τ_2 disconnect at all states or τ_1 and τ_2 partially overlap at some state. An example of obligation property for normal variable is the following formula

$$((x \geq 0) = true)\ or\ (not((y = 0) = false)) \tag{67}$$

which means that either x is nonnegative at all states or y equals zero at some state.

5.4. Response properties

Response property means if A occurs, then B must occur. To specify this property, we firstly create a node as follows:

$$\varepsilon = implies(\alpha, \beta) \tag{68}$$

Then we define response property as following form

$$\varepsilon = true; \tag{69}$$

where α and β are expression of bool type. This formula indicates that if α is true, then β must be true. An example of response property is the following formula

$$\varepsilon = implies((x \geq 0), dc(\tau_1, \tau_2)); \varepsilon = true; \tag{70}$$

which means that if $(x \geq 0)$, then τ_1 and τ_2 must disconnect.

6. Application

Communication based train control system(CBTC) is a typical safety-critical cyber-physical system. It is a new generation of train control system. According to the definition of IEEE, CBTC has three characteristics: (a) high-resolution train location determination, independent of track circuits; (b) continuous, high capacity, bidirectional train-to-wayside data communications; (c) train-borne and wayside processors performing vital functions. A CBTC system usually includes five subsystems, which is shown in Fig.3.

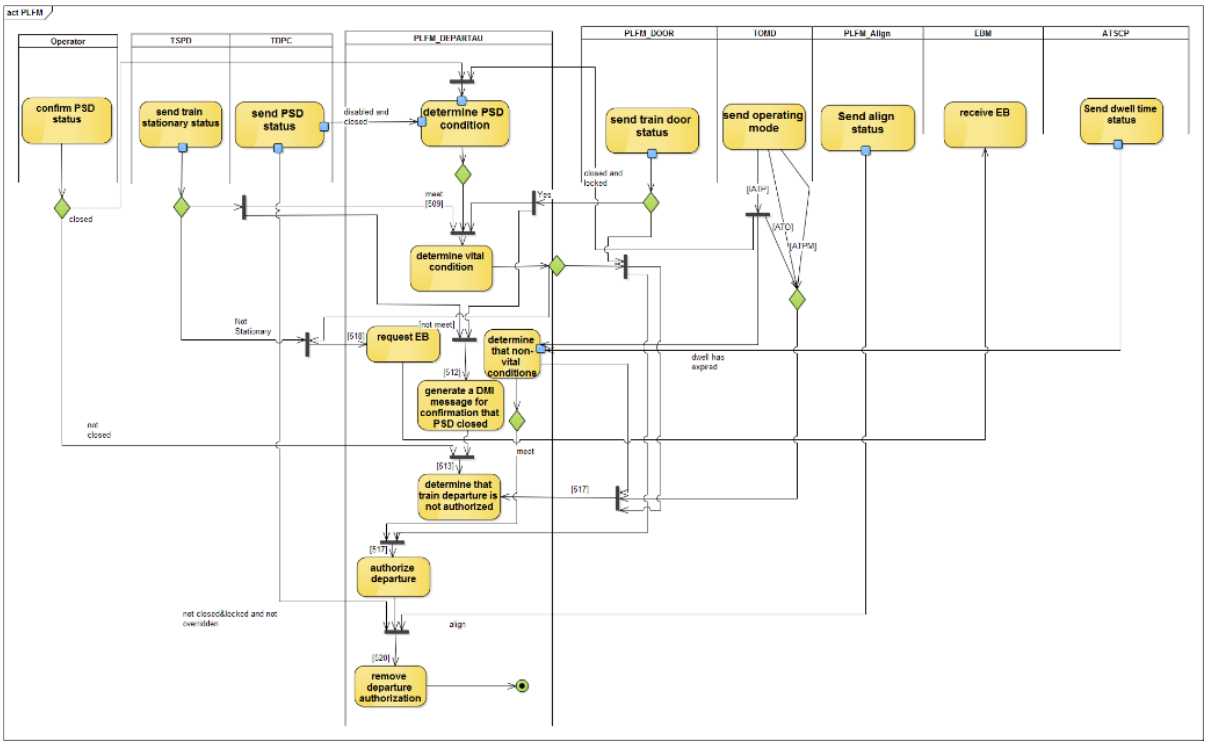

Figure 3. Activity diagram of PLFM

ATS	Automatic Train Supervision subsystem
ATO	Automatic Train Operation
ATP	Automatic Train Protection
ZC	Zone Controller subsystem
TGC	Train Ground Communication subsystem

Figure 4. Subsytems of CBTC

Name	Type	Name	Type
iIsPositioned	*bool*	*iTrainDoorDisable*	*bool*
iPosUncertainty	*int*	*iTrain*	*spatial*
iTrainStationaryState	*int*	*iProPlate*	*spatial*
iTargetPlfSkip	*bool*	*iDetWindow*	*Spatial*
iTimeProPlate	*int*	*iTimePerTime*	*int*
iIsTrnDoorClsLck	*bool*	*oIsTrainAligned*	*bool*
iPlateFormType	*enum*	*oIsSignalActive*	*bool*
iCuPfSkip	*bool*	*oIsAuthorizing*	*bool*

Figure 5. Input/Output variables in PLFM

On Board Control Unit(OBCU) mainly contains ATP and ATO part. Platform Management(PLFM) is an important module in OBCU. The PLFM module mainly performs protection when train is within platform area. The tasks of PLFM include deciding whether the train is aligned, when the door can open and whether the train can be authorized departure. We will introduce some requirements of PLFM in next subsection. And these requirements are related with spatial representation.

6.1. Requirements of platform management

(1) SignalActivation: The platform management(PLFM) shall accept a proximity plate signal as active only if:

- the PLFM detects the proximity plate within the detection window;
- the proximity plate has been detected for a time period equal to or greater than ACTIVE_TIME.

(2) Aligning: The train is aligned if the conditions are met:

- Platform type: real.

- Alignment detection: Proximity plate signal Active
- Train stationary: yes
- Current platform skipped: no
- Train position: Detection zone
- Position lost: no
- Positional uncertainty ≤ UNCERTAINTY_LIMIT
- Time persistency ≥ PERSISTENCY_TIME

(3) Authorizing: When the train is disjoint with the platform area, the PLFM shall determine that vital conditions for enabling propulsion and authorizing departure are met when all of the following conditions are true:

- The train is aligned;
- All train doors are closed and locked;
- All train doors are disabled;

where ACTIVE_TIME, UNCERTAINTY_LIMIT and PERSISTENCY_TIME are three constants. Proximity plate is a hardware equipment installed in the track. The three requirements presented above shows the main process when the train arrives the platform. After proximity plate is active, the train will be considered aligned if some other conditions are met. After the train is aligned, departure will be authorized if other conditions are met. Detailed process of PLFM are presented in Fig.4

6.2. Modeling with ST-LUSTRE

According to the requirements presented before, we firstly define some input/output variables. They are summarized in Fig.5.

(1) The Model of SignalActivation

```
node getSignalActive
      (iDetWindow:Spatial,
       iProPlate: Spatial,
       iTimeProPlate: int)
   returns (oIsSignalActive: bool);
let
   oIsSignalActive =
     (iTimeProPlate>=ACTIVE_TIME)
   when ntpp(iProPlate, iDetWindow)
tel
```

(2) The Model of Aligning

```
node getAligned(iDetWindow:Spatial,
            iProPlate:Spatial,
            iTimeProPlate: int,
            iPlateFormType:int,
            iTrainStationaryState:int,
            iPosUncertainty: int,
            iTimePerTime:int)
     returns (oIsTrainAligned: bool);
let
   isSignalActive = getSignalActive
   (iDetWindow,iProPlate, iTimeProPlate)
  oIsTrainAligned = isSignalActive and
          (iPlateFormType = 0) and
          (iCuPfSkip = false) and
          (iTrainStationaryState = 0)
          and (iIsPositioned = true)
          and (iPosUncertainty =<
          UNCERTAINTY_LIMIT) and
```

```
        ntpp(iTrain, iDetWindow)
        and (iTimePerTime >=
            PERSISTENCY_TIME)
tel
```

(3) The Model of Authorizing

```
node getAuth(iTrain:Spatial,
        iPlfArea:Spatial,
    iTrainStationaryState:int,
    iIsTrnDoorClsLck:bool)
  returns (oIsAuthorizing:bool)
let
    oIsAuthorizing =
    dc(iTrain, iPlfArea) and
    (oIsTrainAligned = true)
    and (iIsTrnDoorClsLck=true)
    (iTrainDoorDisable = true)
Tel
```

6.3. ST-LUSTRE modeling environment

We have implemented a tool which supports modeling and simulation of ST-LUSTRE programs. It is based on an open source project Ptolemy. We add ST Director and ten operators in it: Pre, Followed By, When, Current, dc, ec, eq, po, tpp, ntpp. We also add a new data type named *spatial*. The tool for ST-LUSTRE and the model of PLFM are shown in Fig.6.

In this application, spatial data type is implement in ((x, y, z), r), where (x, y, z) is the point position and r is the radius. The simulation result projecting on time dimension is shown in Fig.7, where the horizontal axis means time and the vertical axis means whether the result is satisfied (0 is false and 1 is true). Fig.8 shows the simulation result projecting on space dimension. The horizontal axis means space(the distance from the starting point) and the vertical axis means the result as mentioned above.

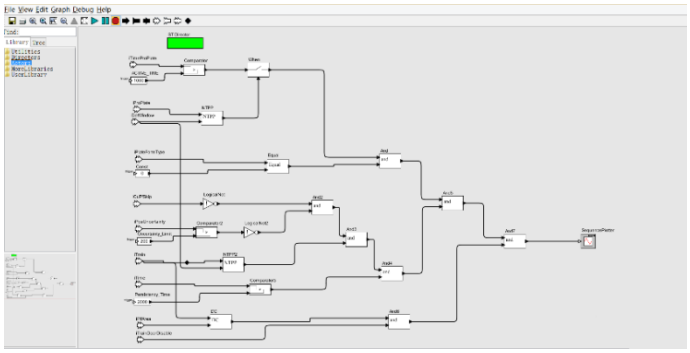

Figure 6. ST-LUSTRE Modeling Environment

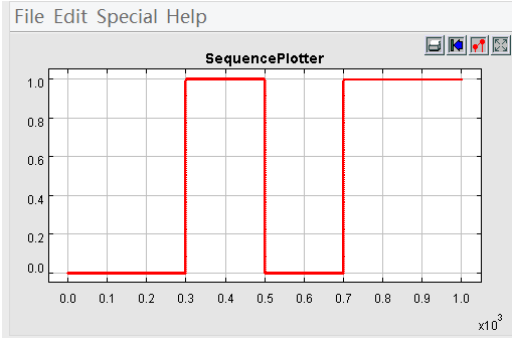

Figure 7. Projections of Simulation Result On Time Dimension

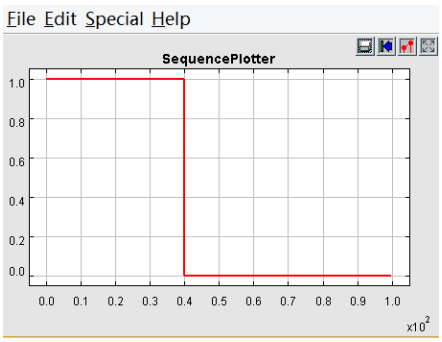

Figure 8. Projection of Simulation Result On Space Dimension

6.4. Property analysis of PLFM

In this part, we express some properties of PLFM. For example, one safety property can be expressed as follows: when train is aligning, for all the states, train door must not open. Formula with ST- Lustre is as follows:

isAligned = getAligned()
verifyResult =
implies(isAligned,
(iIsTrnDoorClsLck = true))
verifyResult = true

Besides, one example of guarantee property can be expressed as follows: when train is authoring, signal is activated at least once.

isAuthoring = getAuth()
isSignalActive = getSignalActive()
verifyResult = isAuthoring and (not
(isSignalActive =false))
verifyResult = true

Guaranteeing system correctness is crucial for safety-critical cyber-physical systems. Here, we specify safety property and guarantee property of PLFM. In other SCCPSs, other properties also can be specified by ST-LUSTRE. If we want to verify those properties, we can use existing model checking tools by transforming spatial related proposition to non-spatial proposition.

7. Related work

Some work has been done in the area of extending spatial logic with others. In [7], the authors extend signal temporal logic with tree spatial superposition logic. In [12], the authors extend the signal temporal logic with spatial operators. However, to the best of our knowledge, synchronous languages have not been extended with spatial logic before, which are widely used in CPSs development. On the other hand, research in spatio-temporal logic have been proposed. In [17], Wolter and Zakharyaschev propose a hierarchy of languages intended for qualitative spatio-temporal representation and reasoning. In [11], Muller presents a formal theory for reasoning about motion of spatial entities in a qualitative framework. In [3], Bennett constructs a two-dimensional logic capable of describing topological relationships that change over time. In application field, in [4], Alberto Del Bimbo et al. applies spatio-temporal logic in symbolic representation and visual query. In [10], Stephan Merz applied spatio-temporal logic in modeling mobile communication system. However, all of those mentioned above do not focus on cyber-physical systems.

Furthermore, some work about extension on synchronous languages has been proposed. In [5], Caspi and Marc extend both operational sematic and clock calculus of LUSTRE. In [1], Kerstin Bauer has considered a synchronous language Quartz, and he extend it to Hybrid Quartz. However, they do not extend synchronous language on space.

In this paper, we propose a novel spatio-temporal language named ST-LUSTRE for safety-critical cyber-physical system. We use a new method to import space into synchronous language. We model systems and specify safety-critical properties with ST-LUSTRE.

8. Conclusions

Safety-critical cyber-physical systems must guarantee behavior correctness to avoid hazards during operation. Formal method is considered as a promising method to ensure correctness for both system models and implementations. However, it still has some challenges because SCCPSs need cooperate with the spatio-temporal dynamics of physical world. To solve this problem, a language is defined as the first step, which can model and specify both temporal and spatial attributes.

A language named ST-LUSTRE is defined in this paper. It can model SCCPSs and specify properties which are related to temporal and spatial features. This language extends the famous synchronous language LUSTRE on the spatial dimension, which is often used in modeling and implementing safety-critical systems in practice. We define spatial type into LUSTRE and six corresponding spatial operators. The semantics of those are defined as the function of time to be

conformed with LUSTRE. Four kinds of typical SCCPSs properties and an engineering application in transportation domain are illustrated in this paper. Furthermore, a tool which supports modeling systems with ST-LUSTRE has been developed. In the future, the study will be focused on how to abstract spatial objects better and related algorithms.

Acknowledgements

This paper is partially supported by the projects funded by the NSFC Key Project 61332008, National Key R&D Program of China 2017YFB1001801, SHEITC160306 and NSFC 61572195.

References

1. K. Bauer, "A new modelling language for cyber-physical systems", *Electronic Journal of Differential Equations*, vol. 136, no. 2, pp. 281-286, 2012
2. B. Bennett, "Spatial reasoning with propositional logics", *Principles of Knowledge Representation & Reasoning*, pp.51-62, 1994
3. B. Bennett, A. G. Cohn, F. Wolter and M. Zakharyaschev, "Multi-dimensional modal logic as a framework for spatio-temporal reasoning", *Applied Intelligence*, vol. 17, no. 3, 239-251, 2002
4. A. D. Bimbo, E. Vicario and D. Zingoni, "Symbolic description and visual querying of image sequences using spatio-temporal logic", *IEEE Transactions on Knowledge & Data Engineering*, vol. 7, no. 4, pp. 609-622, 1995
5. P. Caspi, D. Pilaud, N. Halbwachs and J. A. Plaice, "Lustre: A declarative language for programming synchronous systems", *Symposium on Principles of Programming Languages Acm*, vol. 259, no. 89, pp. 911, 1987
6. P. Caspi and M. Pouzet, "A functional extension to Lustre", *International Symposium on Languages for Intentional Programming*, 1995
7. I. Haghighi, A. Jones, Z. D. Kong, E. Bartocci, R. Gros and C. Belta, "SpaTeL: a novel spatial-temporal logic and its applications to networked systems", *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, 2015
8. N. Halbwachs, P. Caspi, P. Raymond and D. Pilaud, "The synchronous data flow programming language LUSTRE", *Proceedings of the IEEE*, vol. 79, no. 9, pp. 1305-1320, 2000
9. E. A. Lee, "Cyber Physical Systems: Design Challenges", *IEEE International Symposium on Object Oriented Real-Time Distributed Computing*, pp.363-369, 2008
10. S. Merz, M. Wirsing and J. Zappe, "A spatio-temporal logic for the specification and refinement of mobile systems", *In: Fundamental Approaches to Software Engineering, Springer Berlin Heidelberg*, pp. 87-101, 2003
11. P. Muller, "A qualitative theory of motion based on spatio-temporal primitives", *Principles of Knowledge Representation and Reasoning*, pp. 131-141, 1998
12. L. Nenzi, L. Bortolussi, V. Ciancia, M. Loreti and M. Massink, "Qualitative and quantitative monitoring of spatio-temporal properties." *Runtime Verification. Springer International Publishing*, 2015
13. D. A Randell, Z Cui and A. G. Cohn, "A Spatial Logic based on Regions and Connection", *Principles of Knowledge Representation and Reasoning*, pp.165-176, 1992
14. R. Rajkumar, I. Lee, L. Sha and J. Stankovic, "Cyber-physical systems: the next computing revolution", *Design Automation Conference IEEE*, pp. 731-736, 2010
15. L. Sha, S. Gopalakrishnan, X. Liu and Q. X. Wang, "Cyber-Physical Systems: A New Frontier", *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing IEEE Computer Society*, pp.3-13, 2008
16. Y. Tan, M. C Vuran and S. Goddard, "Spatio-Temporal Event Model for Cyber-Physical Systems", *IEEE International Conference on Distributed Computing Systems Workshops*, pp. 44-50, 2009
17. F. Wolter and M. Zakharyaschev, "Spatio-temporal representation and reasoning based on RCC-8", *Principles of Knowledge Representation and Reasoning*, pp. 3-14, 2000