Hybrid Systems in XYZ/E*

YAN An TANG Zhi-song

(Lab. of Computer Science Institute of Software The Chinese Academy of Sciences Beijing 100080)

Abstract Hybrid System is an embedded computing system composed of computers and physical instruments. It allows the inclusion of continuous components in a reactive real-time system. XYZ/E is a temporal logic language based on Manna-Pnueli's Linear Time Temporal Logic. It combines both static and dynamic semantics in a unified framework and supports the whole procedure of stepwise refinement, i.e. from the abstract specifications to executable codes. In this article, the authors first specify and verify hybrid systems with XYZ/E, then introduce the computational model we selected for the hybrid systems and discuss the formal description of hybrid systems in the TLL XYZ/E. Finally the verification methodology in the framework is discussed. Compared with the related work, XYZ/E supports state transitions such that it can specify complex control algorithms in a convenient way.

Key words Hybrid system, phase transition system, temporal logic language, XYZ/E, program specification, verification.

Hybrid System is an embedded computing system composed of computers and physical instruments. It allows the inclusion of continuous components in a reactive real-time system. The computers control and synchronize the physical units' behavior to meet the special requirements for the system. Because of its extensive usage in the industrial and defensive fields, Hybrid System lays the most important emphasis on the safety and reliability properties.

The TLL XYZ/E is outlined in Ref. [1]. It is a temporal logic language based on Manna-Pnueli's Linear Time Temporal Logic. XYZ/E combines both static and dynamic semantics in a unified framework and supports the whole procedure of stepwise refinement, i.e. from the abstract specifications to executable codes.

1 Computational Model

In hybrid systems, the continuous components are characterized by differential equations. Every program state occupies a continuous time interval and all the differential equations activated in this interval prescribe the behaviors of the system. State transitions occur at the discrete time points. The value of variables can be changed and another set of differential equations will be activated.

In XYZ/E, hybrid systems are modeled as phase transition system (PTS). PTS was originally presented in Ref. [2]. We choose it as the computational model because it is formal and precise enough and can be

* This research is supported by the National Natural Science Foundation of China (国家自然科学基金, No. 69673019).

Yan An was born in 1975. He received a B.S. degree in Computer Science from University of Science & Technology of China, and MS degree in Computer Science both from Institute of Software, the Chinese Academy of Sciences, and from Brown University. His research interests are programming language, real-time systems, software engineering and object-oriented database. **Tang Zhi-song** was born in 1925. He is a professor, a doctoral supervisor and an academician. His current research areas include temporal logic language, software engineering, CASE tools and software architecture.

Manuscript received 1997-12-12, accepted 1998-03-30.

conveniently represented in XYZ/E after careful extension.

The following definition is quoted from Ref. [2].

A phase transition system (PTS) $\Phi=\langle V,\Theta,T,A,\Pi\rangle$ consists of:

- $V=\{v_1,v_2,\ldots,v_n\}$: A finite set of system variables. It can be partitioned into the set of discrete variables D and the set of integrators I. A special integrator T is introduced to represent the master clock.
- Θ: The initial condition.
- T: A finite set of transitions. Each transition $\tau\in T$ is a function

$$\tau:\Sigma\mapsto 2^{\Sigma}$$

mapping each state $s\in\Sigma$ into a (possibly empty) set of τ-successor states $\tau(s)\subseteq\Sigma$.

The successor function for τ is defined by a transition relation $\rho_\tau(V,V')$, which may refer to V and modify $V-\{T\}$. For every $\tau\in T$, it is required that $\rho_\tau\rightarrow T'=T$.

- A: A finite set of activities. Each activity $\alpha\in A$ is represented by an activity relation:

$$p_\alpha\rightarrow I(t)=F^{\alpha}(V^0,t)$$

where p_α is a predicate over D called the activation condition of α. The vector equation $I(t)=F^{\alpha}(V^0,t)$ defines the evolution of the integrators throughout a phase of continuous change according to the activity α.

- Π: The time-progress condition specifying a global restriction over the progress of time.

2 Specify Hybrid Systems in XYZ/E

2.1 Specify state transitions

In XYZ/E, we choose PTS as the computational model and use the iterative method to realize it in executable codes. Suppose the sampling interval Δ satisfies the precision requirement for the system.

Referring to the length of the temporal logic operator @ as "$|@|$", we define the Next-time Operator Series NEXT $=\{\$O_1,\$O_2,\ldots\}$. It is required that $|\$O_i|>|\$O_{i+1}|$, and for all $\varepsilon>0$, $\exists N>0$, $\forall n>N$, $|\$O_n|<\varepsilon$. Based on this, we introduce a new operator $\$O_\varepsilon$ called ε-next-time operator, which means: Next time, and $|\$O_\varepsilon|<\varepsilon$. Because for all ε, $\exists N>0,\forall n>N$, $|\$O_n|<\varepsilon$, if we define $\$O_\varepsilon=\$O_{N+1}$ then $|\$O_\varepsilon|<\varepsilon$. In other words, we may think of the operator $\$O_\varepsilon$ as an element of NEXT and it is picked out based on the time upper bound ε.

The PTS model requires that for every $\tau\in T$, $\rho_\tau\rightarrow T'=T$. This kind of assertion is valid in computational models but in no temporal logic languages. It is illogical if $T'=T$ and $V'\neq V$, because the program variable cannot have two different values at the same time. We have to introduce a new operator to represent these transitions.

First of all, we can find a constant ε for the given system, which guarantees that we can omit the time occupied by the operators whose length is less than ε and the correctness of the program is not affected, e.g. we can define $\varepsilon=\Delta/1000$ where Δ is the sampling interval. Based on this ε, we can pick out the ε-next-time operator $\$O_\varepsilon$ from the operator set NEXT for the next time operators which is expected to be zero-time-consuming. For example, the transition $\tau\in T$ in PTS can be described as:

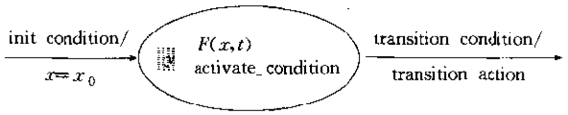
$$LB=y\wedge Enable\_condition(\tau)\Rightarrow\$O_\varepsilon(\rho_\tau(V,V')\wedge LB=z).$$

Please note that in the language level $T'=T+|\$O_\varepsilon|$ and in the model level $T'=T$. This difference does not affect the consistency between the model and the program because we have supposed that $|\$O_\varepsilon|<\varepsilon$.

2.2 Specify hybrid systems

Let's take a look into the following hybrid system:

The system enters state y after initialization and the activity characterized by activate_condition$\rightarrow F(x,t)$ is activated in state y. When the assertion "transition_condition" is satisfied the transition_action is enabled and the system enters the next state y'. The system variable t refers to how long the system has been in the current state.



We specify this system in XYZ/E as below:

$LB=y\wedge x=x_0\Rightarrow$

$\quad<>(transition\_condition\wedge \$O_\epsilon(transition\_action\wedge LB=y'))$

WHERE $(LB=y\wedge activate\_condition\rightarrow F(x,t))$

$\{x=transition\_action(F1(x_0,t))\}$ /* *post_condition*, where $F1(x_0,t)$ is the value of x before the transition and it satisfies the diff. eq. $F(x,t)$ */

It specifies that when the pre_condition is $x=x_0$, and the activation condition is true, the value of x satisfies the differential equation $F(x,t)$ until a transition is enabled. The program will take the transition and enter the next state y'.

The executable codes for this specification are given below with the iterative method:

```
{$x=x_0$}   /* pre_condition */
$LB=y\rightarrow \$O_\epsilon(t=0\wedge last\_t=0\wedge LB=y1)$;
$LB=y1\wedge activate\_condition\Rightarrow$
        $\$O<0,\Delta>(x=x+(t-last\_t)*F'(x,t)\wedge last\_t=t\wedge LB=y1)$;
$LB=y1\wedge transition\_condition\Rightarrow$
        $\$O_\epsilon(transition\_action\wedge LB=y')$;
{$x=transition\_action\ (F1(x_0,t))$}   /* post_condition */
```

Here "$\$O<0,\Delta>$" means: Next time, and the time interval is less than Δ.

The first conditional element sets the temporary variable t and $last\_t$ as 0 and starts the iteration. The second conditional element specifies that if the predicate activate_condition is satisfied the program will take an iteration within Δ time. The iterative equation $x_{i+1}=x_i+(t_{i+1}-t_i)*F'(x_i,t_i)$ is deducted from $F(x,t)$. The value of t is set by the system when the conditional element is fired. The temporary variable $last\_t$ is introduced to record the last value of t.

The third conditional element specifies if the predicate transition_condition becomes true, the system will take the transition and enter the next state.

We have used the following assumptions in the above discussion:

- All the conditional elements are fired as soon as their condition is satisfied, i.e. "$LB=y\wedge R\Rightarrow\ldots$" is put into action as soon as the FOL R is satisfied.
- The sampling interval Δ is short enough to satisfy the precision requirement.
- The differential equation is suitable for the iterative method.

Example 1. Let's take a look at the following hybrid system.

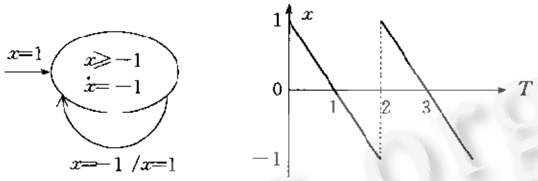
Its corresponding PTS Φ is:

$$V=I:\{x,T\}$$
$$\Theta:x=1\wedge T=0$$
$$T:\{\tau\}\ \text{WHERE}\ \rho_\tau:x=-1\wedge x'=1\wedge T'=T$$
$$A:\{\alpha\}\ \text{which is specified as}\ \$T\rightarrow x=x^0-t$$
$$\Pi:x\geqslant -1$$

We specify this hybrid system in XYZ/E as below:

$$LB = y \wedge x = 1 \Rightarrow$$
$$<> (x = -1 \wedge \$O_\epsilon(x = 1 \wedge LB = y))$$
$$\text{WHERE } \Box(LB = y \wedge x > -1 \rightarrow \dot{x} = -1)$$

Please note that if p_a is the activation condition of the activity α in the PTS model, then we define *activate_condition*$= p_a \wedge p_\Pi$, where p_Π is the time progress restriction related to α in Π. As to this example, we have activate_condition$= \$T \wedge x \geqslant -1$, i.e. $x \geqslant -1$.



Now we refine it into the executable codes. Deducting from the original description $x > -1 \rightarrow \dot{x} = -1$, we have the iteration equation $x_{i+1} = x_i + (t_{i+1} - t_i) * (-1)$. The executable program is shown below:

$\{x=1\}$ /* pre condition */

$LB = y \Rightarrow \$O_\epsilon(t=0 \wedge last\_t=0 \wedge LB = y1)$;

$LB = y1 \wedge x > -1 \Rightarrow \$O<0,\Delta>(x = x-(t-last\_t) \wedge last\_t = t \wedge LB = y1)$;

$LB = y1 \wedge x \leqslant -1 \Rightarrow \$O_\epsilon(x=1 \wedge LB = y)$;

$\{x=1\}$ /* post_condition */

Please note that the firing condition in the third conditional element is $x \leqslant -1$, not $x = -1$. It is the special need in the iterative method because despite the precision assumption, it is possible that in some program runs $x = -1 - \varepsilon$, where ε is a neglectable positive value.

2.3 The benefits of the multi-control mechanism for hybrid systems

XYZ/E realizes the dynamic semantics by introducing the concept of program control. Every control occupies a virtual processor. LB is used to represent the program control and $LBi = z$ represents that program control i is in state z. The multi-control mechanism is allowed in XYZ/E and every program control has its own set of states. We define the state set of the system Σ as $\Sigma = \Sigma_1 \times \Sigma_2 \times \ldots \times \Sigma_r$, here Σ_i, $i = 1..r$, is the state set of control i.

In the single-control model, exactly one activity is activated in a given system state. This activity encloses behaviors of all the integrators and is always described by a group of differential equations. In the multi-control model, every integrator is assigned to a specific control and each control deals with one (several) differential equation(s). The program is thus decomposed into several simpler processes.

Example 2. The activity $\begin{pmatrix} \dot{x} = F1(x,y,t) \\ \dot{y} = F2(x,y,t) \end{pmatrix}$ can be specified in the single-control and multi-control models as below:

(1) In the single-control model

Specification:

$$LB = y \wedge x = x_0 \wedge y = y_0 \Rightarrow$$
$$<> (transition\_condition \wedge \$O_\epsilon(transition\_action \wedge LB = y'))$$
$$\text{WHERE } \Box(LB = y \wedge activate\_condition \rightarrow (\dot{x} = F1(x,y,t) \wedge \dot{y} = F2(x,y,t)))$$

$\{(x,y) = transition\_action(F1'(x_0,y_0,t), F2'(x_0,y_0,t))\}$

/* post_condition, where $(F1'(x_0,y_0,t), F2'(x_0,y_0,t))$ are the values of (x,y) before the transition and they satisfy the diff. eq. $\begin{pmatrix} \dot{x} = f1(x,y,t) \\ \dot{y} = F2(x,y,t) \end{pmatrix}$ */

Executable code:

```
    {$x = x_0 \wedge y = y_0$}   /* pre_condition */
    $LB = y \Rightarrow \$O_t(t = 0 \wedge last\_t = 0 \wedge LB = y1)$;
    $LB = y1 \wedge activate\_condition \Rightarrow$
      $\$O < 0,\Delta > (x = x + (t - last\_t) * F1(x,y,t) \wedge$
            $y = y + (t - last\_t) * F2(x,y,t) \wedge last\_t = t \wedge LB = y1)$;
    $LB = y1 \wedge transition\_condition \Rightarrow \$O_t(\text{transition\_action} \wedge LB = y')$;
    {$(x,y) = \text{transition\_action}(F1'(x_0,y_0,t),F2'(x_0,y_0,t))$}   /* post_condition */
```

All the integrators must be handled together in the single-control model. This work may be exhausting if the number of integrators grows large.

(2) In the multi-control model

The integrators can be handled separately in the multi-control model. All the controls synchronize with each other by the communication command. In our discussion we always use the following assumption: The message transmission consumes no time.

Specification:

```
    $LB_x = y1 \wedge x = x_0 \Rightarrow$
        $<>(transition\_condition_x \wedge \$O_t(transition\_action_x \wedge LB = y1'))$;
    $LB_y = y2 \wedge y = y_0 \Rightarrow$
        $<>(transition\_conditon_y \wedge \$O_t(transition\_action_y \wedge LB = y2'))$
    WHERE $\square(LB_x = y1 \wedge activate\_condition_x \rightarrow \dot{x} = F1(x,y,t)) \wedge$
            $\square(LB_y = y2 \wedge activate\_condition_y \rightarrow \dot{y} = F2(x,y,t))$
    {$x = transition\_action_x(F1'(x_0,y_0,t)) \wedge y = \text{transition\_action}_y(F2'(x_0,y_0,t))$}
```

Executable code:

• The program control of x:

```
{$x=x_0$}                                                    /* pre_condition */
$LB_x=y1 \Rightarrow \$O_t(t_x=0 \wedge last\_t_x=0 \wedge LB=y11)$;
$LB_x=y11 \wedge ch_{yx}? \Rightarrow \$O_t(ch_{yx}?\ y \wedge LB=y11)$;       /* get the value of $y$ */
$LB_x=y11 \wedge \text{activate\_condition}_x \Rightarrow$
    $\$O<0,\Delta>(x=x+(t_x-last\_t_x)*F1(x,y,t) \wedge$       /* iteration */
        $ch_{xy}!\ x \wedge$                                  /* send the value of $x$ */
        $last\_t_x=t_x \wedge LB_x=y11)$;
$LB_x=y11 \wedge transition\_condition_x \Rightarrow$
  $\$O_t(transition\_action_x \wedge ch_{xy}!\ x \wedge LB_x=y1')$;    /* transition */
{$x=transition\_action_x(F1'(x_0,y_0,t))$}                   /* post_condition */
```

• The program control of y:

```
{$y=y_0$}                                                    /* pre_condition */
$LB_y=y2 \Rightarrow \$O_t(t_y=0 \wedge last\_t_y=0 \wedge LB=y21)$;
$LB_y=y21 \wedge ch_{xy}? \Rightarrow \$O_t(ch_{xy}?\ x \wedge LB_y=y21)$;     /* get the value of $x$ */
$LB_y=y21 \wedge \text{activate\_condition}_y \Rightarrow$
  $\$O<0,\Delta>(y=y+(t_y-last\_t_y)*F2(x,y,t) \wedge$         /* iteration */
```

$ch_{yx}!\ y \wedge$ /* send the value of y */

$last\_t_y = t_y \wedge LB_y = y21$);

$LB_y = y21 \wedge transition\_condition_y \Rightarrow$

$\$O_t(transition\_action_y \wedge ch_{yx}!\ y \wedge LB_y = y2')$; /* transition */

$\{y = transition\_action_y(F2'(x_0, y_0, t))\}$ /* post_condition */

If the communication is reliable and efficient, and there is no restriction on the iteration sequence, the representations in single-control model and multi-control model are consistent and the sets of program states are of the same size. On the other side, the state set of each control, i.e. the state set facing the programmers, is much smaller than that of the whole program. So programs in multi-control model are more concise and legible than those in single-control mode.

2.4 The lasting conditions

Normally there are three firing conditions.

Figure 1 is the most common situation. The conditional element is activated as soon as the firing condition becomes true. Figure 3 is a special case in hybrid systems in which the firing condition is a cumulating integrator, e.g. the leak of gas. This can be conveniently expressed because we have discussed how to specify integrators in XYZ/E. However, for those cases like Fig. 2 which require the condition to remain TRUE for a period, a specific command is needed.

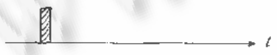


Fig. 1 On-Point condition

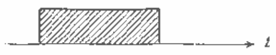


Fig. 2 Lasting condition

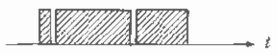


Fig. 3 Cumulating condition

We introduce a new operator "$\$O\{l,u\}$" meaning: Next time, which is within u time, and the firing condition must last for at least l time. It is used in the condition element in the following way:

$$LB = y \wedge R \Rightarrow \$O\{l,u\}(Q \wedge LB = z)$$

It is defined as:

$$LB = y \wedge R \Rightarrow \$O(DELAY(l) \wedge LB = y')\ \underline{\$W}(\neg R \wedge \underline{\$O}LB = y)$$

$$LB = y' \Rightarrow (Q \wedge LB = z)\ \underline{\$W}(OVER(u-l) \wedge \underline{\$O}LB = exception\ \$V\ \neg R \wedge \underline{\$O}LB = y)$$

Here $\underline{\$O}$, $\underline{\$W}$ are the NEXT-TIME and UNLESS operators in the model level.

3 Verifying Hybrid System in XYZ/E

Reference [2] discussed the verification of PTS. In that article, activities and transitions are combined to form the set of extended transitions T_H, i.e. let $\Phi: \langle V, \Theta, T, A, \Pi\rangle$, then $T_H = T \vee T_\Phi$, where $T_\Phi = \{\tau_\alpha | \alpha \in A\}$. For each $\alpha \in A$, the transition relation of τ_α is given by:

$$\rho_{\tau_\alpha}: \exists\ \Delta > 0(D' = D \wedge p_\alpha \wedge I' = F^\alpha(V, \Delta) \wedge \forall\ t \in [0, \Delta). \Pi(D, F^\alpha(V, t)))$$

Then the verification of safety properties can use the following rule:

1. $\Theta \rightarrow p$
2. $\underline{\{p\}\tau\{p\}\ \text{for every}\ \tau \in T_H}$

$\Phi \models \Box p$

However, the length of the basic transition "$\$O$" is not zero in a temporal logic language such as the TLL XYZ/E. We cannot verify the validity of p when the next-time operator $\$O$ is in processing because there is no place for the programmers to specify what happens during these intervals. What we can verify by the above rule is the validity of p outside the time occupied by $\$O$.

Because the safety property in a hybrid system means "for all $t \geqslant 0$, p holds", we must take all the time.

including the intervals occupied by $\$O$, into account. In the linear time temporal logic, all we can do is to discuss the system's behaviors on the discrete series of time points, i.e. $t_0, t_1, \ldots$ (t_{i+1} is the next time of t_i). We call this kind of safety which is related to the length of the next operator $\$O$ as "Δ safety" and note it as "$\Box_\Delta$". It can be verified by the following induction rule:

1. If p holds at t_0;
2. for all t_i, $p \rightarrow \$O(p)$;
3. then $\Box_\Delta p$.

4 Related Work

In this article we have concentrated on the computational model, the specification, the implementation and verification of hybrid systems in the TLL XYZ/E. This work is further discussed in Refs. [3,4]. As an example for the work in this article, Ref. [3] gives the specification, stepwise refinements and verification of the Steam-boiler Control Specification problem[4] and Ref. [5] gives the final implementation of this problem. Reference [4] also illustrated XYZ/E's capacity for visual programming. Compared with the related work, i.e. Duration Calculus[6], XYZ/E supports state transitions such that it can specify complex control algorithms in a convenient way.

References

1 Tang C S. A temporal logic oriented toward software engineering—— an introduction to XYZ system (I). Chinese Journal of Advanced Software Research, 1994,1(1):1~29

2 Zohar Manna, Amir Pnueli. Clocked transition systems, Logic & Software Engineering. In: Pnueli A, Lin H eds. Proceedings of a workshop 1995. Beijing, Singapore: World Scientific, 1996

3 Yan An, Tang C S. A unified linear-time temporal logic solution to the steam-boiler control specification problem. Technical Report, No. ISCAS-LCS-97-05, Laboratory of Computer Science, Institute of Software, the Chinese Academy of Sciences, 1997

4 Abrial J-R. Steam-boiler control specification problem. The Dagstuhl Meeting. Method for Semantics and Specification, 1994

5 Yan An, Tang C S. Building hybrid real-time systems in XYZ/E——the final implementation of the steam-boiler control specification problem. Technical Report, No. ISCAS-LCS-97-06, Laboratory of Computer Science, Institute of Software, the Chinese Academy of Sciences, 1997

6 Zhou Chao-chen, Wang Ji, Ravn A P. A formal description of hybrid systems. Technical Report, ISCAS-LCS-94-14, 1994

基于 XYZ/E 的混成系统

阎安 唐稚松

(中国科学院软件研究所计算机科学开放研究实验室 北京 100080)

摘要 混成系统是由计算机和物理设备组成的嵌入式实时计算系统.它允许在交互式实时系统中引入连续变化的单元.XYZ/E 是基于 Manna-Pnueli 的线性时序逻辑的程序设计语言.它将程序的动态语义与静态语义统一在同一框架中,支持从抽象的程序规范到可执行代码的逐步求精的全过程.该文使用 XYZ/E 语言描述和验证混成系统.首先介绍了计算模型,然后介绍了 XYZ 语言对混成系统的形式化描述,最后介绍了混成系统的验证.与同类工作相比,XYZ/E 支持状态转换,从而可以方便地描述复杂的控制算法.

关键词 混成系统,相位转换系统,时序逻辑语言,XYZ/E,程序规范,验证.

中图法分类号 TP311