ISSN1000-9825 Journal of Software 软件学报 2000,11(8):996~1002

基于 XYZ/RE 的多媒体同步器自动构造方法*

赵琛[1] 唐稚松[1] 马华东[2]

[1](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)

[2](北京邮电大学计算机学院 北京 100088)

E-mail: zc@ox.ios.ac.cn/mhd@bupt.edu.cn

摘要 XYZ 系统是一个以增强软件可靠性和提高软件生产率为目的的程序开发支撑系统.它由时序逻辑语言(temporal logic language,简称 TLL)XYZ/E 和以该语言为基础的一组软件工程工具组成.为了研究 XYZ 系统在多媒体领域中的应用问题,介绍了一种依据多媒体对象时序描述自动生成用 XYZ/RE 表示的播放同步器的方法.XYZ/RE 是时序逻辑语言族 XYZ/E 中表示实时系统的子语言.与相关工作比较,该方法不仅可以处理简单的时序关系,而且可以处理嵌套的时序关系,所产生的同步器可以复用于不同的节目.

关键词 时序关系,XYZ 系统,XYZ/RE,同步器.

中图法分类号 TP316

自动程序设计是人们在软件工程研究领域中所追求的理想目标之一,即希望根据程序的需求可自动地获取一个程序,而这个程序正好达到此需求.不幸的是,这种理想主义最终被证明是不可行的.那么,程序自动化技术研究什么呢?这一技术主要在 3 个方面研究语言之间的自动转换[1].第 1 个方向是基于数学方法的程序自动转换,研究以数学的等价变换方式实现由函数式语言程序到高级语言程序,或函数式语言程序相互之间的自动转换.第 2 个研究方向是在程序设计的各个环节上借助人工智能方法来代替人的工作,即在人们设计程序的各个环节上均试图用人工智能方法实现的专家系统来完成.第 3 个方向是编译的编译,提高编译自动化水平的研究即所谓编译的编译(compiler compiler).编译系统中语法分解方面的自动化技术已经较为成熟,所以剩下应研究的问题在其语义方面.这就是通常所说的语义导引的编译(semantic directed compiler)这方面的研究.上述程序自动化技术的研究都是希望探索一种通用规范语言之间或规范语言到算法语言之间的自动转换,难度很大.

本文将要讨论的是从多媒体时序描述,即 Allen 的区间代数[2],到 XYZ/RE[3]表示的播放系统之间的自动转换.本文第 1 节介绍一种分布式多媒体播放系统的体系结构.第 2 节是 XYZ[1]系统和时序逻辑语言簇 XYZ/E[3]的一个简单介绍.第 3 节介绍时序描述语言.第 4 节讨论媒体同步器的自动生成方法.第 5 节是与相关工作的比较.最后是总结.

1 播放器体系结构

图 1 是一个分布式多媒体播放系统体系结构.为了使问题简单化,便于集中研究播放的同步控制,把媒体间的同步和数据在分布式网络上的传输分发分开考虑,而且假定所有媒体单位在网络传输时不会丢失,这个假设可以由网络的质量来保证.下面将分别简要介绍各个部分的结构和功能.从图 1 可以看出,播放器包括这样几个控制实体:流控制协议(flow-control protocol,简称 FCP)、媒体播放控制协议(media-phone control protocol,简

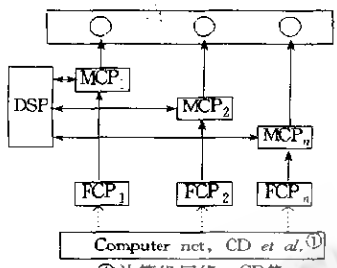
* 本文研究得到国家"九五"重点科技攻关项目基金(No. 98-780-01-07-01)和国家 863 高科技项目基金(No. 863 306-ZT02-04-1)资助.作者赵琛,1967 年生,博士,副研究员,主要研究领域为软件工程.唐稚松,1925 年生,研究员,博士生导师,中国科学院院士.主要研究领域为计算机科学,软件工程.马华东,1964 年生,博士,副教授,主要研究领域为多媒体技术,计算机动画和图形学.

本文通讯联系人:赵琛,北京 100080,中国科学院软件研究所计算机科学开放研究实验室

本文 2000-01-17 收到原稿,2000-04-04 收到修改稿

称MCP)和目标同步协议(destination synchronization protocol,简称DSP).

FCP负责控制多媒体数据流从计算机网络或CD等设备上的接收,主要用硬件实现;MCP负责控制多媒体数据流在实际播放单元上的播发,一般也主要用硬件来实现.多媒体的时序同步关系包括两个方面:媒体对象内部的同步和媒体之间的同步.MCP也控制着多媒体数据流的内部同步,媒体对象内部的同步不是本文的研究重点,但也是同步的一部分,这里只作以下简单说明.



①计算机网络,CD等.

Fig. 1 The architecture of a broadcasting system
图1 播放器的体系结构

假定一个节目有K个媒体数据流(stream).STM_1,STM_2,...,STM_k.每一个媒体数据流又分成n个数据单位,即$STM_i=STM_i(1),STM_i(2),...,STM_i(n)$,$1\leqslant i\leqslant k$.

由于网络的阻塞导致数据传输的延迟,解决这个问题的方法是在播放单元上重复播放刚才播放的数据单位一个单位时间,也就是用原来的播放数据单位代替了迟到的数据单位,等到迟到的数据单位到来时,跳过该数据单位,接着播放下一个数据单位.这种内部的控制主要由MCP来实现.而且不同的多媒体对象所采用的策略可能是不同的,对于字符流可以允许时序同步不必像动画那么严格,但是不能采取丢弃的策略,因为字符丢失,消息就不全了.对于动画和声音,一个多媒体流所丢弃的数据单位不能太多,而且不能出现两个连续的丢弃.考察多媒体节目的播放质量,主要考虑两个参数.一个是媒体单位丢弃率.一般来说,对于视频,只要不出现两个连续的媒体单位被丢弃,对播放的效果影响就不大.直观上说,如果两个或两个以上的连续媒体单位由于传输延迟而被丢弃,在播放时就会出现较大的抖动现象.另一个需要考虑的参数是相对应的媒体单位之间的偏差不能超过一个确定值.比如,视频和音频之间的偏差不能大于80ms.

DSP负责控制多媒体数据流之间的同步是本文主要研究的问题,我们也将DSP称为同步器.

2 XYZ系统和XYZ/E简介

2.1 系统概述

XYZ系统是一个基于时序逻辑[4]的软件工程系统.它包含一种时序逻辑语言XYZ/E及实现3类软件开发方法的软件工程工具.其中,XYZ/E既是一个线性时间的时序逻辑系统,又是兼具通常程序语言风格且可以用统一的形式框架同时表示程序规范及各种可执行程序机制的程序语言.实际上,XYZ/E是基于Manna-Pnueli线性时序逻辑的系列化语言族(Xiliehua Yuyan Zu),它相当于一种广谱语言(wide spectrum language).其中,各子语言分别表示不同的程序设计方式或程序范型(paradigms),故XYZ/E是以一致的逻辑框架来表示的统一范型.我们将在下面分别介绍与本文工作相关的成分.

XYZ/E语言的基本命令称之为条件元CE(conditional element),它的形式为

$$LB=S_i\wedge P=>\$O(v_1,...,v_k)=(e_1,...,e_k)\wedge\$OLB=S_j, \tag{1}$$

$$LB=S_i\wedge P=>@(Q\wedge LB=S_j). \tag{2}$$

这里,"=>"表示蕴含;@是一个时序逻辑算子;P,Q是两个一阶逻辑合式公式,分别称为一个条件元的条件部分和动作部分;LB是一个指明当前状态和下一状态的控制元.

通信命令包括输入命令和输出命令,其表示形式如下:

$$LB_i=y\wedge R=>\$OChNm?\ x\wedge\$OLB_i=w, \tag{3}$$

$$LB_i=y\wedge R=>\$OChNm!\ z\wedge\$OLB_i=w. \tag{4}$$

这里,LB_i表示第i个进程的控制变量;"$ChNm?\ x$"与"$ChNm!\ z$"分别表示两个谓词(比如写成?$(ChNm,x)$,!$(ChNm,z)$),即"由通道$ChNm$接收信息送入变量x中"和"由通道$ChNm$送出表达式z的值".根据通信方式(同步或异步),式(3)与式(4)可分别看成是几个不同条件元令的缩写,见文献[1].

2.2 实时语言 XYZ/RE

为了表示实时计算,需先引入表示时钟性质的常谓词 $DELAY(d)$及 $OVER(m)$,它们分别表示"已延迟 d 时刻"及"自进入本状态以来执行时间已超过 m 时刻".这两个谓词的真假是由程序以外的环境中的时钟决定的.用这两个谓词即可将 $\$O\{l,u\}$,$\Diamond\{l,u\}$,$\Box\{l,u\}$,$\$U\{l,u\}$,$\$W\{l,u\}$等实时时序算子,在原来的模型中予以定义[1].令@$\{l,u\}$表示上述前3种实时时序算子中的任意一种,则任一如下的包含实时时序算子@$\{l,u\}$的条件元为

$$LB=y\wedge R=\!>@\{l,u\}(Q\wedge LB=z), \tag{5}$$

可解释(或定义)(当@$\{l,u\}$为 $\$O\{l,u\}$时)为如下不含实时时序算子的条件元:

$$\begin{aligned} &LB=y\wedge R-\!>\$O(DELAY(l)\wedge LB=y')\ \underline{\$W}(\sim R\wedge \underline{\$OLB=y});\\ &LB=y'=\!>\$O(Q\wedge LB=z),\\ &\underline{\$W}(OVER(u-l)\wedge \underline{\$OLB=\text{exception}}-y'\ \$V\sim R\wedge \underline{\$OLB}-y). \end{aligned} \tag{6}$$

事实上,式(6)正好表示了式(5)(当@$\{l,u\}$为 $\$O\{l,u\}$时)的语义.式(6)中"$\sim R\wedge \underline{\$OLB=y}$"表示此条件元中的条件 R 必须在全过程中一直保持为真,否则,返回原来的位置.至于是否在第(1)条命令后要加入这一判断,则可能因问题的要求不同而有不同的取舍.

现在考虑通信命令的下、上时限问题.并发系统有同步和异步两种基本的通信方式,在XYZ/E中都能表示,而且它们的语法是一样的.在XYZ/E中实际上是表示两谓词,即"?$(ChNm,x)$"与"!$(ChNm,y)$".当它们在条件元中出现时,相当于动作部分(即条件式(1)中的Q).故对于通信命令说,其实时限制并非直接表示在这些命令之中,而应考虑未加缩写的情况条件元.详见文献[1].对于发送命令,由于是异步方式,发送时间基本上是计算要发送的表达式的时间.为了考虑简化,我们假定发送命令的执行时间为0.即发送的表达式仅是常量或一个变量.实时模型的形式讨论见文献[1,3].

2.3 并行语句和选择语句

XYZ/E语言能够方便地表示并发程序和不确定程序.下面简单介绍两个基本的语句:并行语句和选择语句.

一组进程组成一个并发进程是由并行语句来表示的,其形式如下:

$$LB=y_i\wedge R=\!>\|[ProsInstNmi_1(Pari_1);\ldots;ProsInstNmi_k(Pari_k)].$$

这里,$ProsInstNmi_j$即是由进程实例化命令所引入的进程实例名字,它代表了相应的进程实例.不同的进程实例可以是由同一进程文本在不同进程实例化命令中所生成的不同实例.

在一组进程或动作中随机选取一个执行用选择语句来表示,其形式如下:

$$LB=y_i\wedge R=\!>!!\ [Cond_1|\!>ExeAct_1;\ldots;Cond_k|\!>ExeAct_k].$$

其中 $Cond_i$ 与 $ExeAct_i(i=1,\ldots,k)$分别表示条件与动作,它们都是一阶逻辑公式(在 $ExeAct_i$ 中可包括赋值或控制等式).有关并行语句和选择语句的形式定义见文献[1].

3 时序描述语言

为了在多媒体时序描述中应用逐步求精和程序自动生成等形式化技术,这里,我们研究一个表达能力足够的Allen建立区间代数(interval algebra)的子集合作为多媒体对象时序关系的描述语言.

```
program::=temporalrelation (program,program)|term|identifier
term::=temporalrelation(strring,strring)|term,term
temporalrelation::=before before duration|finish|overlap duration|overlap|
                   start|during duration meet|equal
identifier::=letter {(letter|digit)}|identifier identifier
duration::=empty|[float,float]
float::=unsignedinteger.[unsignedinterger]
```

unsignedinteger;=digit{digit}

数字和字母的定义与其他形式语言的定义相同.这里忽略.

4 同步器构造

4.1 示 例

首先用一个例子直观地说明本文构造媒体同步器的基本思想.

例 1:用XYZ/RE构造多媒体节目$(before(before(s_1,s_2)^{[1,2]},overlap(s_1,s_3)),s_4),finish((before(s_1,s_2)^{[1,2]},overlap(s_1,s_3)),s_5)$的同步器.

为了构造例1的同步器的XYZ/RE表示,首先给由多媒体流s_1,s_2和s_3构成的复合流一个名字s.

controls(%chan synch,commch,parch,seqch)==[

$LB=l_0=>\$O$ synchc?^ $\$OLB-l_1$;/*接到同步信号*/

$LB=l_1=>\$O$ commch!command^ $\$OLB=l_2$;/*授权*/

$LB=l_2=>\$O$ parch!^ $\$OLB=l_3$;/*向并行对象发同步信号*/

$LB-l_3->\$O$ commch?^ $\$OLB=l_4$;/*接收结束信号*/

$LB=l_4=?\$O$ {1,2}seqch!^ $\$OLB=l_5$;/*向顺序对象发同步信号*/

$LB=l_5=>\$OLB$=STOP]

plays(%chan commch,mcpch)==[

$LB=l_0=?\|[controls_1,plays_1,controls_2,plays_2,controls_3,plays_3]$;

$LB=l_2=>\$O$ commch!END=l_3;/*汇报播放结束*/

$LB=l_3=>\$OLB$=STOP]

$controls_4$(%chan synch,commch)==[

$LB=l_0=>\$O$ synchc?^ $\$OLB=l_1$;/*接到同步信号*/

$LB=l_1=>\$O$ commch!command^ $\$OLB-l_2$;/*授权*/

$LB=l_2->\$OLB-$STOP]

$plays_4$(%chan commch,mcpch)==[

$LB=l_0=>\$O$ commch?^ $\$OLB-l_1$;/*接到播放授权*/

$LB-l_1->\$O$ mcpch?^ $\$OLB=l_2$;/*接到播放请求*/

$LB-l_2->\$O$ mcpch!^ $\$OLB=l_3$;/*回应播放请求*/

$LB=l_3=>\$OLB$=STOP]

$controls_5$(%chan synch,commch)==[

$LB=l_0=>\$O$ synchc?^ $\$OLB=l_1$;/*接到同步信号*/

$LB=l_1=>\$O$ commch!command^ $\$OLB=l_2$;/*授权*/

$LB=l_2=>\$OLB$=STOP]

$plays_5$(%chan commch,mcpch)==[

$LB=l_0=>\$O$ commch?^ $\$OLB=l_1$;/*接到播放授权*/

$LB=l_1=>\$O$ mcpch?^ $\$OLB=l_2$;/*接到播放请求*/

$LB=l_2=>\$O$ mcpch!^ $\$OLB=l_3$;/*回应播放请求*/

$LB=l_3=>\$OLB$=STOP]

控制的基本思路是用两个XYZ/RE进程控制一个媒体对象的播放,一个控制同步,一个控制与MCP的连接,即播放.媒体对象之间的同步关系通过进程之间的通信来实现.采用进程之间相互通信激活的方法,而不是用绝对的时间轴在绝对的时刻激活进程,可以保证动态的同步关系,从而使媒体播放得光滑,保证质量,这在分布式系统中尤为重要.

4.2 自动生成算法

根据上述例子的基本思路,我们研究通用的构造算法.在一个多媒体节目的时序描述中,有直接时序关系的多媒体对象称为相关多媒体对象.对于一个流 s,根据它与其他多媒体流之间的时序关系,把它的相关多媒体对象又分为以下几种.

(1) s 的同步控制进程要从这些流的控制进程中接收同步信号的;

(2) s 启动之后,s 的控制进程要向这些流的控制进程发送开始同步信号的;

(3) s 启动之后,s 的控制进程要向这些流的控制进程发送开始同步信号并等待回应的;

(4) s 结束之前,s 的控制进程要向这些流的控制进程发送结束同步信号的;

(5) s 结束之后,s 的控制进程要向这些流的控制进程发送同步信号的.

下面分别将上述相关多媒体对象严格定义为前同步相关对象、强开始同步相关对象、弱开始同步相关对象、结束同步相关对象和后同步相关对象这几个集合.

定义1. 前同步相关对象

$$Front(s)=\{s_i \mid before(s_i,s),meet(s_i,s),overlap(s_i,s)\in SP,during(s,s_i)\in SP,start(s_i,s)\in SP,equal(s_i,s)\}.$$

定义2. 强开始同步相关对象

$$StrBegin_1(s)=\{s_i \mid start(s,s_i)\in SP,equal(s,s_i)\in SP\},$$

$$StrBegin_2(s)=\{s_i \mid start(s_i,s)\in SP,equal(s_i,s)\in SP\}.$$

定义3. 弱开始同步相关对象

$$Begin(s)=\{s_i \mid during(s_i,s)\in SP,overlap(s,s_i)\in SP\}.$$

定义4. 结束同步相关对象

$$End_1(s)=\{s_i \mid finish(s,s_i)\in SP,equal(s,s_i)\in SP\},$$

$$End_2(s)=\{s_i \mid finish(s_i,s)\in SP,equal(s_i,s)\in SP\}.$$

定义5. 后同步相关对象

$$Behind(s)=\{s_i \mid before(s,s_i)\in SP,meet(s,s_i)\in SP\}.$$

算法1. controls 构造算法

输入:时序描述中一个流 s 的前同步相关对象、强开始同步相关对象、弱开始同步相关对象、结束同步相关对象和后同步相关对象;

输出:XYZ/RE 表示流 s 的同步控制进程.

(1) if $Front(s)=\varnothing$

then 生成一个接收从系统初始激活信号的条件元;

else 生成 $|Front(s)|$ 个接收从 s_i 的 $controls_i$ 进程传来的同步信号的条件元,即

$LB=$当前标号 $->!!\lceil fch_1?\triangleright \$OLB=l_1V'fch_2\triangleright ?\$OLB=l_1\ldots fch_k\triangleright ?\$OLB=l_1\rceil$;

$LB=l_1->!!\lceil fch_1?\triangleright \$OLB=l_2V'fch_2\triangleright ?\$OLB=l_2\ldots fch_k?\triangleright \$OLB=l_2\rceil$;

$\vdots$

$LB=l_k->!!\lceil fch_1?\triangleright \$OLB=NEXTV'\ldots fch_2?\triangleright \$OLB=NEXT\ldots fch_k?\triangleright \$OLB=NEXT\rceil$;

(2) 生成一个接收从 plays 进程传来的请求信号的条件元;

(3) 生成 $|StrBegin_1(s)|$ 个向 $StrBegin_1(s)$ 中的流的对应的控制进程发送强开始同步信号的条件元;

(4) 生成 $|StrBegin_1(s)|$ 个接收从 $StrBegin_1(s)$ 中的流的对应的控制进程传来的强开始同步回应信号的条件元,即

$LB=$当前标号 $->!!\lceil sch_1?\triangleright \$OLB=l_1V'ch_2?\triangleright \$OLB=l_1\ldots sch_k?\triangleright \$OLB=l_1\rceil$;

$LB=l_1->!!\lceil sch_1?\triangleright \$OLB=l_2V'sch_2?\triangleright \$OLB=l_2\ldots sch_k?\triangleright \$OLB=l_2\rceil$;

$\vdots$

$LB=l_k->!!\lceil sch_1?\triangleright \$OLB=NEXTV'\ldots sch_2?\triangleright \$OLB=NEXT\ldots sch_k?\triangleright \$OLB=NEXT\rceil$;

(4') 生成 $|StrBegin_2(s)|$ 个回应强开始同步的条件元;

(5) 生成一个向 plays 进程发送回应信号的条件元;

(6) 生成|*Begin*(*s*)|个向 *Begin*(*s*)中的流的对应的控制进程发送弱开始同步信号的条件元;

(7) 生成一个接收从 plays 进程传来的结束请求的条件元;

(8) 生成|$End_1(s)$|个向 $End_1(s)$中的流的对应的控制进程发送结束同步信号的条件元;

(9) 生成|$End_1(s)$|个接收从 $End_1(s)$中的流的对应的控制进程传来的结束同步回应信号的条件元,即

$LB=$当前标号$=>!![ech_1?\triangleright \$OLB=l_1V'ech_2?\triangleright \$OLB=l_1 \ldots ech_k?\triangleright \$OLB=l_1]$;

$LB=l_1=>!![ech_1?\triangleright \$OLB=l_2V'ech_2?\triangleright \$OLB=l_2 \ldots ech_k?\triangleright \$OLB=l_2]$;

$\vdots$

$LB=l_k=>!![ech_1?> \$OLB=NEXTV' \ldots ech_2?\triangleright \$OLB=NEXT \ldots ech_k?\triangleright \$OLB=NEXT]$;

(9′) 生成|$End_2(s)$ 个回应结束同步的条件元;

(10) 生成一个向 plays 进程发送同意结束信号的条件元;

(11) 生成|*Behind*(*s*)|个向 *Behind*(*s*)中的流的对应的控制进程发送同步信号的条件元.

算法2. plays 构造算法

输入:时序描述中一个流 s 的前同步相关对象、强开始同步相关对象、弱开始同步相关对象、结束同步相关对象和后同步相关对象;

输出:XYZ/RE 表示流 s 的播放进程.

(1) 生成一个接收从 controls 进程传来的播放授权的条件元;

(2) if s 是一个嵌套结构,生成一个并行语句播放结构中的对象,转(8);

(3) 生成一个接收从 MCP 传来的播放请求的条件元;

(4) 生成一个向 MCP 发送允许播放信号的条件元;

(5) 生成一个接收从 controls 进程传来的允许结束授权的条件元;

(6) 生成一个接收从 MCP 传来的结束请求的条件元;

(7) 生成一个向 MCP 发送允许结束信号的条件元;

(8) 生成一个进程结束的条件元.

5 与相关工作的比较

最早讨论同步器自动构造工作的是文献[5],应该说这种思想很有创新性,但由于历史的原因也存在很大的局限性.与这项工作相比,本文讨论的自动构造方法进行了以下的扩充和改进:

(1) 构造方法面向可组合的时序描述,为软件、硬件的复用奠定了基础;文献[5]的构造方法是面向单层时序描述的.

(2) 构造算法生成的是可执行的时序逻辑语言 XYZ/RE,因此,算法得到的是一个可实际运行的控制系统;文献[5]构造的仅仅是一个时间自动机;

(3) 将多媒体流的内部同步控制和多媒体流之间的同步控制严格区分开,便于实现高质量的、灵活的播放控制;文献[5]却没有区分这两种同步机制.

(4) 多媒体流之间的控制方式也根据时序不同而分为5种基本的同步方式:前同步、强开始同步、弱开始同步、结束同步和后同步,一般不需要给出多媒体流的实际长度.文献[5]中没有区分不同的同步方式,所以,需要给出多媒体流的实际长度.这也是它无法处理嵌套结构的时序描述的一个主要原因,因为即使多媒体流的长度是给定的,由于有 during,overlap 等时序关系的存在,一个节目的长度也是不确定的.

最新的多媒体同步集成语言 SMIL(synchronized multimedia integration language)[6]通过浏览器来实现节目的同步集成,因此,它只支持最简单的两种时序描述,即并发和顺序.用〈par〉和〈seq〉来指明多媒体对象的时序关系.

6 总 结

本文主要介绍了多媒体同步器的自动构造算法.依据一个多媒体节目的时序描述,可以自动地得到一个用XYZ/RE实现的同步器,能够实现这种自动化的主要原因是多媒体系统的特殊性和具体性.由一个节目的时序描述,我们就可以知道应该如何构造相应的同步器.对于普通问题的时序描述,这种自动化技术是很困难或不可能的.

参考文献

1 Tang Zhi-song *et al*. Temporal Logic Programming and Software Engineering (Vol. 1). Beijing: The Science Press, 1999 (唐稚松等.时序逻辑程序设计与软件工程(上册).北京:科学出版社,1999)

2 Allen J F. Maintaining knowledge about temporal intervals. Communications of the ACM, 1986,26(11):832~843

3 Tang Zhi-song, Yang Li. Real time programming with XYZ System. Chinese Journal of Advanced Software Research, 1995,2(4):317~325

4 Manna Z, Pnueli A. The Temporal Logic of Reactive and Concurrent Systems: Specification. New York: Springer-Verlag, 1992

5 Kshirasagar Naik. Specification and synthesis of a multimedia synchronizer. IEEE Journal of Selected Areas in Communications, 1994,12(5):544~549

6 The World Web Wide Constrium. http://www.w3.org

A Methodology for Automatically Constructing a Multimedia Synchronizer in XYZ/RE

ZHAO Chen[1] TANG Zhi-song[1] MA Hua dong[2]

[1](*Laboratory of Computer Science Institute of Software The Chinese Academy of Sciences Beijing* 100080)

[2](*School of Computer Engineering Beijing University of Posts and Telecommunications Beijing* 100088)

Abstract XYZ system is a programming support system with the goal to enhance reliability and productivity of software development. It consists of a TLL (temporal logic language) XYZ/E to serve as its kernel and a suite of software engineering tools. In order to study the application of XYZ system to the multimedia field, in this paper, the authors present a method for automatically generating synchronizer in XYZ/RE from the temporal specification of a multimedia program. XYZ/RE is a sub-language representing real time system in XYZ/E. Compared with the related work, this method can transform not only a simple temporal specification but also a nested temporal specification, so that the generated synchronizer can be reused for different purposes.

Key words Temporal relations, XYZ system, XYZ/RE, synchronizer.