文章编号: 1001-0920(2005)06-0616-05

ICSL: 一种用于仿真实体智能化控制的脚本语言

胡圣明, 李青山, 陈 平
(西安电子科技大学 软件工程研究所, 陕西 西安 710071)

摘 要: 研究一种用于仿真实体智能化控制的脚本语言ICSL (Intelligent Control Scripting Language). 分析领域特征, 将语言的语义划分为领域相关语义和领域无关语义; 对领域相关语义进行编码, 结合形式化描述方法和构造工具, 定义ICSL 并实现其解释器; 通过解释器将ICSL 语言描述的智能化控制策略翻译为仿真系统中实体的行为序列, 为用户提供了一种灵活表达控制策略的手段. 此外, ICSL 还采用元数据控制策略实现了多个侧面的扩充机制, 从而解决了语言系统难以定制和演化的问题.

关键词: 智能化控制; 仿真; 脚本语言; 控制策略; 领域相关语义
中图分类号: TP311 **文献标识码**: A

ICSL: A scripting language used for intelligent control on simulation entity

HU Sheng-ming, LI Qing-shan, CHEN Ping
(Software Engineering Institute, Xidian University, Xi'an 710071, China. Correspondent: HU Sheng-ming, E-mail: shmhu@mail.xidian.edu.cn)

Abstract: A scripting language, intelligent control scripting language (ICSL), used for intelligent control is presented. Semantics of ICSL is divided into domain specific semantics and domain independent semantics after analyzing domain features. By encoding the semantics of simulation fields and using formal describing methods, a precise definition of ICSL is given and its interpreter is constructed. ICSL presents intelligent strategy because of its interpreter ability to translate the intelligent strategy into the sequence of entity behaviors in a simulation system. Also, metadata control mechanism is employed to extend the semantics of ICSL and solve the problem of customizing a language.

Key words: intelligent control; simulation; scripting language; control strategy; domain-specific semantics

1 引 言

智能化是一种感知环境、作出决定和控制行为的能力[1]. 仿真系统中存在大量的实体, 这些实体应具有自动的智能化行为. 对仿真实体的控制通常采用人工控制方式和直接编码方式. 采用前者所能达到的智能化程度最高, 因为感知环境、作出决定和控制行为都是由人来完成, 但其缺点在于控制不灵活且效率低, 而且对每一个实体都采用人工控制的方式也不现实; 若采用后者, 则控制效率较高, 但无法变更对仿真实体的控制策略, 使实体不具备自动的智能化行为.

针对上述问题, 研究一种智能化控制语言来控制仿真实体是十分必要的. 由于仿真系统的不断演化, 控制语言本身应具备很强的可扩充性. 本文将控制策略与仿真系统相分离, 定义了一种用于描述智能化控制策略的语言(ICSL), 并实现了该语言的解释器. 基于ICSL 语言, 策略制定者可描述出智能化控制策略, 进而, ICSL 解释器将这种控制策略映射

收稿日期: 2004-08-29; **修回日期**: 2004-12-07.
基金项目: 国家自然科学基金项目(60473063); 国家教育部博士点基金项目(20030701009); "十五"国防预研项目(41306060106).
作者简介: 胡圣明(1979—), 男, 陕西西安人, 博士生, 从事程序设计语言、逆向工程等研究; 陈平(1953—), 男, 陕西西安人, 教授, 博士生导师, 从事电子信息系统软件开发、面向对象技术等研究.

到仿真实体的行为变化逻辑中, 便可达到对仿真实体智能化控制的目的. 此外, 针对语言系统难以定制的问题, ICSL 系统借助元数据控制机制, 使得 ICSL 语言具有较强的扩充性, 从而解决了语言系统的定制和自身演化问题.

2　ICSL 语言的设计

ICSL 语言系统根据控制策略对信息数据进行计算, 再将计算结果映射到仿真系统中, 仿真实体根据该策略控制要求来产生新的行为逻辑. 为便于表现智能化控制策略的控制流程, ICSL 定义了基本的控制语句; 为达到获取仿真系统的环境信息和控制仿真实体行为的目的, ICSL 语言支持从仿真环境中抽象出各种核心对象, 以便策略制定者使用; 为支持领域特有的一些常量和处理方法, ICSL 语言提供了面向领域特征的内置常量和内置函数; 为适应领域需求的不断变化, ICSL 语言还提供了多个侧面的扩充机制.

2.1　ICSL 语言的词法与语法定义

作为一种语言, ICSL 有其自身的一套标记符号. 表 1 给出了部分记号的正则表达式. 为方便定义, 首先给出一些辅助记号以便后续定义引用:

LETTER = [a-zA-Z], DIGIT = [0-9].

表 1　ICSL 语言的记号定义

类型	类型标识	正则表达式
关键字与运算符	…	If, float, while, +, {, [, …
标识符	ID	LETTER(LETTER|DIGIT)*
字符常量	C_CHAR	"[^\"]*"
十进制整形常量	C_DINT	(DIGIT)+
时间常量	C_TME (C_DINT:)? (C_DINT:C_DINT)	
⋮		

本文在设计 ICSL 的词法时充分考虑效率和扩展性, 将不易变化的部分采用正则表达式描述, 并利用词法分析工具自动生成词法分析器, 以在扫描完单词后便得到记号的类型, 提高分析的效率; 而对于需要扩展的部分, 可将其归入标识符, 再由识别得到的标识符串判断其记号类型, 以获得较高的扩展性.

ICSL 语言中的内置常量就是在词法分析阶段识别, 将内置常量的词法识别模式归纳为 ID, 当识别出 ID 后, 再从一个内置常量表中查询以确定识别出的字符串是否是一个常量. 如 TRUE, FALSE, PI 等基本常量, Hostile (表明实体是敌方), FixWing (表明实体是固定型机翼飞机) 等领域特征常量以及各种环境因素常量.

ICSL 语言的语法采用上下文无关文法 G(Vt, Vn, S, P) 描述, 其中开始符号 S = ICSL. 一个完整的 ICSL 脚本分为两部分: 初始化部分和控制部分. 初始化部分主要完成仿真实体的初始化设置任务, 而控制部分则主要控制仿真实体的行为逻辑. 下面的产生式 (1) 描述了 ICSL 语言脚本的结构, 产生式 (3) 描述了 ICSL 语言的语句分类 (所有产生式中大写符号表示非终结符, 小写符号表示终结符):

ICSL → init_section{STATEMENT_LIST}rec_section{STATEMENT_LIST}end_scpt　(1)

STATEMENT_LIST → STATEMENT_LIST STATEMENT \| STATEMENT　(2)

STATEMENT → end_stmt　(3)

\| EXPRESSION_STATEMENT end_stmt

\| IF_STATEMENT end_stmt

\| WHILE_STATEMENT end_stmt

\| TME_STATEMENT end_stmt

\| FOR_STATEMENT end_stmt

\| ⋮

其中 EXPRESSION_STATEMENT 描述了 ICSL 语言中的表达式. 下面的产生式 (4) 和 (5) 描述了表达式中的对象的操作:

EXPRESSION_STATEMENT → EXPRESSION \| …　(4)

EXPRESSION → IDENTIFIER dot ATTRIBUTE　(5)

\| IDENTIFIER dot FUNCTION

\| …

其中: IDENTIFIER dot ATTRIBUTE 表示了对象属性的操作, IDENTIFIER dot FUNCTION 则表示了对象方法的操作. 如表达式 entity. fire() 表示当前脚本控制的实体开火; system. activate (radar) 表示打开雷达设备. 这里的 entity 和 system 为 ICSL 语言中可操作的对象; 这些对象称之为核心对象; fire 和 activate 则为对象上可操作的方法. 后面将详细介绍对象属性、对象方法以及新对象的扩展机制.

时间语句 TME_STETEMENT 是 ICSL 语言的一个特点, 能够描述与时间相关的策略. 产生式 (6) 和 (7) 描述了其结构:

TME_STATEMENT → at open_par TME_EXP close_par do STATEMENT_LIST enddo　(6)

TME_EXP → const_time　(7)

时间语句的一个典型的例子: At (00: 13: 20: 30) do entity. fire() enddo, 表示在第 13min20s30ms 开火.

ICSL 语言语法还定义了各种其他的数学表达式、控制语句及对象操作. 表达式包括常量和变量表达式、函数调用、算术表达式、布尔表达式等; 语句类型则包含了变量定义语句、赋值语句、条件语句和循

环语句等

2.2 ICSL 的语义设计

语言的一种实施可以看作是语言语义的一种定义[2]. ICSL 的语义可分为两个方面: 领域无关语义和领域相关语义. 领域无关语义是指语言本身所固有的, 不局限于任何一种或几种特定领域的语义, 这种语义主要表现为各种表达式的计算及控制语句的解释, 是一种数学计算行为; 领域相关语义则是和某一具体领域相关联, 表示领域内的各种行为的含义. 领域相关语义比领域无关语义的抽象层次更高, 更接近领域问题. 在 ICSL 语言中, 提供了领域语义编码机制来支持语言的移植性和扩充性.

2.2.1 领域相关语义与领域无关语义

ICSL 的领域无关语义与现在各种应用广泛的通用语言是相同的, 如C/C++, java 等. 对领域无关语义的解释是建立在C++ 语言之上, 其原因是C++ 语言的运行效率较高, 且 ICSL 语言的语法和C++ 的语法相似, 采用C++ 实现 ICSL 的解释器较为方便.

ICSL 的领域相关语义集中表现为语言支持的各种核心对象, 核心对象的属性和方法都是针对领域问题进行一定的抽象而得出的. ICSL 语言的使用者通过操纵这些核心对象来表达领域逻辑(控制策略). 图1 比较了C++ 与 ICSL 解决领域问题时的抽象层次.

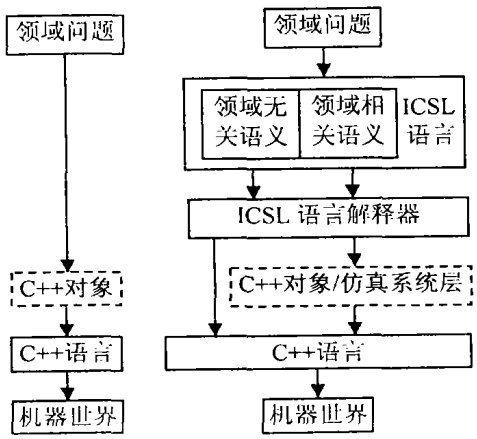

图1 C++ 与 ICSL 的抽象层次

可以看出, ICSL 语言是建立在对象层次之上, 这就表明了 ICSL 语言更接近领域问题. ICSL 的领域无关语义是由C++ 语言直接描述并解释; 而领域相关语义则由C++ 构建出的对象来描述并解释. C++ 语言解决领域问题不仅要构建出对象, 还要描述对象之间的关系; 而ICSL 语言则重在描述对象间的逻辑关系. 根据脚本语言的定义, 脚本语言是为将一组功能强大的组件"粘合"起来而设计的一种语言[3], 所以 ICSL 也是脚本语言的一种.

ICSL 语言中领域相关语义由各种核心对象体现, 核心对象由 ICSL 语言解释器翻译成C++ 对象或仿真系统的行为序列. 在不同的仿真环境下, 核心对象可代表不同的含义, 具有不同的属性和不同的方法, 这种动态可配置的核心对象是在领域语义编码机制的基础上实现的.

2.2.2 领域语义编码

领域语义编码是对领域内各操作的一种抽象, 一个领域语义编码代表了用户的一种意图. 例如: 在仿真环境下, 用户操作 ICSL 核心对象是希望从仿真系统中获取信息, 或使仿真系统的状态发生变化. ICSL 语言首先将用户的各种操作转化为ICSL 内部的语义编码, 然后ICSL 语言解释器将语义编码发送到仿真系统中, 由仿真系统对编码作出响应.

通过语义编码, 使得核心对象属性和方法的处理变得统一: 不论用户是对核心对象的属性还是方法进行操作, 都被 ICSL 解释器转化为一个编码. 事实上, 核心对象的属性和方法只是逻辑上的属性和方法, 属性并不对应真正的物理存储空间, 方法也不直接控制仿真系统中的实体, 语义编码是ICSL 语言系统和仿真系统的唯一接口. 图2 显示了 ICSL 语言的语义编码和仿真系统的关系, 环境参数是根据语义编码所返回的仿真系统中的信息数据.

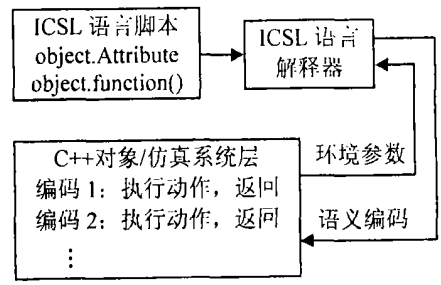

图2 语义编码与仿真系统的关系

ICSL 解释器只需完成属性和方法到语义编码的转换, 仿真系统则关心语义编码与动作序列之间的关系. 语义编码将仿真系统和ICSL 语言系统充分地隔离开, 将两者耦合性降到最低. 而且不同的仿真系统可以对语义编码作出不同的解释, 即一套编码可用于各种不同的仿真系统之上, 即便在一个仿真系统内部, 不同类型的实体也可对同一语义编码作出不同的响应, 充分实现了语义编码的可重用性. 另外, 由于语义编码是可扩充的, 用户可自定义语言的核心对象以及对象的属性和方法.

3 ICSL 的扩充机制

扩充机制是ICSL 语言最重要的特点之一. 除基本的类型、运算和控制外, ICSL 语言的其他成分都

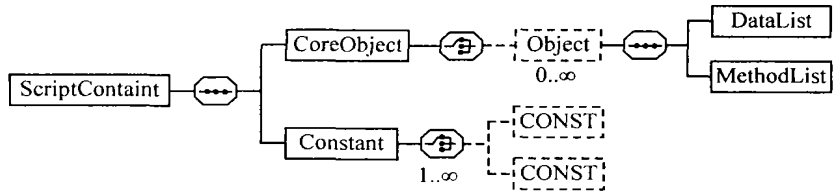

图3　ICSL 元数据文件模式

是可扩充的, 包括: 内置函数和内置常量、核心对象及其属性和方法、语义编码. 在ICSL 语言中, 扩充机制是通过元数据控制策略来实现的. 元数据在程序中不是被加工的对象, 而是用来对程序的运行起控制作用, 并且可以通过值的改变而改变程序的行为[4]. ICSL 存储了需要扩充成分的所有元数据, 通过对元数据的修改来实现语言的扩充. 元数据既是程序行为的一种抽象, 又是高于语言自身数据类型和用户自定义数据类型的一种数据抽象. 基于这种抽象, 元数据仅在系统扩充维护时可能发生修改, 相对稳定. 元数据一般静态存储, 仅在运行中以解释方式控制程序行为, 影响程序的动态行为[5]. 图 3 为ICSL 语言元数据模式图.

图中, CoreObject 表示ICSL 能够支持的核心对象列表(CoreObject 中的Object 表示核心对象), Constant 描述了内置常量的列表, DataList 标签中描述的是核心对象所具有的属性以及属性的类型、名称和读写标志等, MethodList 描述了核心对象的方法信息.

3.1　内置函数和内置常量的扩充

在使用ICSL 语言编写代码的过程中, 可以借用大量已有的库函数. 增加新的库函数只需用户提供函数的接口描述和对应的目标代码, 接口描述是为了让ICSL 系统能够识别此函数, 而目标代码则是为了在运行过程中能正确地调用函数. 各个领域内有很多的常用数值, 这些数值很少发生变化, ICSL 语言通过内置常量来提供支持, 内置常量的扩充只需对相应的内置常量表加以修改即可. 对内置函数的扩充, 用户需修改元数据描述文件, 并在核心对象 buildin 的响应代码中调用此函数, 然后重新进行链接操作; 而对内置常量的扩充则只需修改元数据描述文件.

3.2　核心对象的扩充

核心对象的扩充分为两个方面: 已存在核心对象的修改和增加新核心对象. 核心对象的属性和方法是基于语言内部领域语义编码的, 核心对象属性和方法的操作统一翻译成内部语义编码. 因此, 增加一个新的属性和新的方法需要用户给出属性和方法的元数据描述, 如属性的名称和方法的原型等, 然后指定属性或方法到语义编码的映射即可.

ICSL 语言允许用户自定义核心对象, 包括每个核心对象的属性表、方法表以及属性、方法到语义编码的映射表. 所以新增一个核心对象只需在元数据文件中描述对象初始化时所需加载的各种表数据即可, 而无需做任何编码工作. ICSL 系统会根据元数据文件自动加载核心对象的属性和方法.

3.3　领域语义编码的扩充

领域语义编码是ICSL 语言的核心, 它抽象了用户使用脚本语言的意图. ICSL 语言定义了领域语义的编码, 但并不提供对应的语义动作, 语义动作由ICSL 语言所控制的仿真系统来完成. 如果现有的语义编码无法表示新的语义动作, 或者完全不符合用户要求, 那么用户可在语义编码表中增加新的编码或完全替换现有的各种编码, 从而完成了语义的扩充. 正是这一点可让ICSL 语言应用于各种仿真领域及语言的演化.

4　ICSL 解释器的实现

ICSL 语言的解释器模块组成如图 4 所示. 图中第①部分为准备阶段, 主要完成脚本的编写及合法性检查; 第②部分为运行阶段, 主要任务是将策略应用到仿真系统中. ICSL 脚本采用 ICSL 语言编写策略, 用来控制仿真实体的行为; 元数据文件存储的是关于语言的元信息描述; ICSL 中间表示是经过词法、语法分析后的一种内部表示, 意在提高运行时效率. 各个模块的功能如表 2 所示.

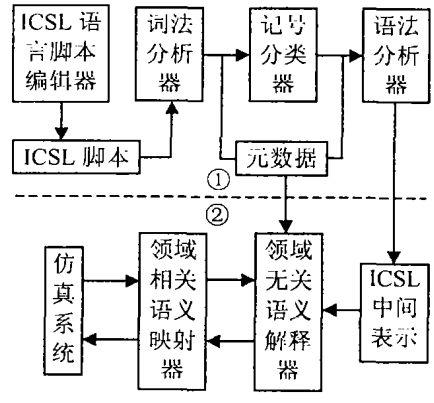

图4　ICSL 语言解释器模块

表2 模块功能

模 块	主要功能
脚本编辑器	脚本编辑程序, 提供各种智能化编辑支持
词法分析器	分析脚本源文件, 输出记号
记号分类器	根据元数据文件描述信息对记号分类
语法分析器	分析脚本程序语法结构, 并生成一种中间表示
领域无关语义解释器	计算领域无关语义
领域无关语义映射器	向仿真系统发送领域语义指令(语义编码), 并提供语言领域语义无关解释器计算所需的领域数据

5 实验研究

ICSL 语言的词法分析器和语法分析器已利用 XDFLEX 和XDYACC 编译工具实现, 记号分类器、语言语义解释器和领域语义映射器已采用C++ 语言实现. 下面是ICSL 系统应用在一个战场仿真环境的样例代码. 其中: system, track, opponent 等为核心对象; activate(), range 等为核心对象方法和属性; radar_sensor, tracking_missile 等则为核心常量.

```
INITIALIZATION-SECTION
    system.activate(radar_sensor);
    //控制仿真实体打开雷达探测设备
REACTION-CONTROL-SECTION
    track.cycle_on(detected_platforms);
    //探测可能对自身产生威胁的实体
    while(track.next()>0) do
        if((track.range<=5 000.0) and
        (track.ident==hostile)) then
        //若实体是敌方并距自身小于5 000 m
        opponent.assign(track.index);
        //锁定此实体作为攻击目标
            if((track.type==land) or
            (track.type==site))
            then //判断此实体类型若是地面实体或站点
            weapon.launch(tracking_missile,
            "rocket");
            //发射代号为"rocket"的跟踪导弹
            endif;
        endif;
    endwhile;
END-SCRIPT
```

下面是仿真系统中相应的响应伪代码.

```
switch(语义编码)
    case 打开雷达设备: 打开雷达设备break;
    case 探测: 探测特定类型实体break;
    ⋮
    case 发射: 发射指定武器break;
    default: break;
```

语义响应代码可位于仿真系统中实体类层次树中的不同层次, 父类可完成子类共同的语义动作, 子类可实现自身特殊的语义动作, 并且子类可覆盖父类的响应行为, 充分地实现语义行为和语义编码的重用. 在配置文件中增加一个新的核心对象, 那么ICSL 语言立即支持此核心对象.

6 结 语

本文针对智能化控制, 研制了 ICSL 脚本语言, 利用XDFLEX 和XDYACC 工具实现了 ICSL 语言的解释器. ICSL 语言系统的特点在于: 1) 提出了领域无关和相关语义, 并针对领域特征对领域语义进行编码; 2) 通过元数据的控制策略, 提供了 ICSL 语言的词法、语法和语义的扩充机制; 3) 语义的定义与实现相分离的结构实现了 ICSL 语言在仿真平台上的移植性.

下一步工作的重点是提供一套扩充工具. 虽然 ICSL 语言提供了很强的扩充机制, 但用户手动扩充 ICSL 容易造成语言不一致性, 用一套工具来实现语言的扩充能有效地避免扩充中可能产生的各种错误, 从而保证语言的有效性和一致性.

参考文献(References)

[1] Panos Antsaklis. Defining intelligent control[R/OL]. IEEE Control Systems Society, 1993, http://citeseer.ist.psu.edu/169851.html.

[2] 周巢尘. 形式语义学引论[J]. *计算机研究与发展*, 1985, 12(7): 1-65.
(Zhou C C. An instruction to formal semantics[J]. *J of Computer Research and Development*, 1985, 12(7): 1-65.)

[3] John K. Ousterhout scripting: Higher-level programming for the 21st century[J]. *IEEE Computer Society*, 1998, 31(3): 23-30.

[4] 李青山, 陈平, 褚华. 支持柔性机制的元数据模型的研究与应用[J]. *西安电子科技大学学报*, 2002, 29(3): 319-323.
(Li Q S, Chen P, Chu H. Research on and the application of the metadata-driven model supporting flexibility[J]. *J of Xidian University*, 2002, 29(3): 319-323.)

[5] Dean E Bushey, Brian A Malloy. A study of dynamic traffic re-routing in the national airspace system[A]. *Proc of the IEEE Annual Simulation Symposium*[C]. Atlanta: IEEE Press, 1997: 104-113.

(下转第624 页)

练, 核函数为高斯径向基核, $p = 1, C = 0.1$, 得到 $R = 20.4, D_{max} = 20.64, D_{min} = 20.3$. 选取 $f = 2, \sigma = 0.001$, 模糊隶属度函数为

$$\mu_i = \begin{cases} 8.81D^2(x_i) + 363.62D(x_i) + 3751.88, \\ \quad 20.3 \leq D(x_i) \leq 20.4; \\ -2.97D(x_i) + 61.25, \\ \quad 20.3 \leq D(x_i) \leq 20.4 \end{cases} \tag{12}$$

得到模糊隶属度函数之后再用 FLS-SVM 进行训练, 采用与 LS-SVM 相同的核函数和模型参数 $(2^{5.75}, 2^{3.5})$, 训练完后对 250 个样本进行预测, 结果如图 3 所示, 预测均方差为 0.072 2.

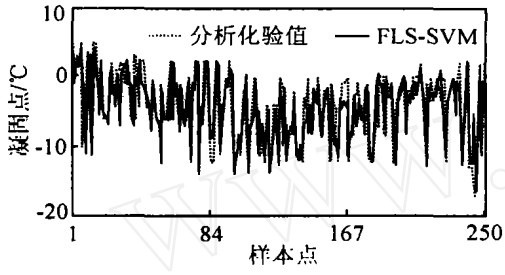

图 3 采用 FLS-SVM 的轻柴油凝固点预测值曲线

由图 2 和图 3 可以看出, FLS-SVM 的拟合效果要好于 LS-SVM, 且预测均方差显著减小.

5 结 语

本文将模糊隶属度概念引入 LS-SVM 中, 提出了一种基于支持向量数据域描述的模糊隶属度函数模型. 首先得到训练集中样本的数据域描述模型, 然后根据样本偏离数据域的程度赋予不同的隶属度. 该方法提高了 LS-SVM 的抗噪声能力, 尤其适合于未能完全揭示输入样本特性的情况. 将提出的方法运用于催化裂化分馏塔轻柴油凝固点的软测量建模, 仿真结果表明, 提出的模糊隶属度函数模型可有效地提高 LS-SVM 的预测精度.

参考文献(References)

[1] 徐敏, 俞金寿. 软测量技术[J]. *石油化工自动化*, 1998, 10(2): 1-3.
(Xu M, Yu J S. Technology of soft sensor [J]. *Automation in Petro-chemical Industry*, 1998, 10(2): 1-3.)

[2] Vapnik V. *Statistical learning theory* [M]. New York: Wiley Springer, 1998: 146-175.

[3] Suykens J A K, Vandewalle J. Least squares support vector machine classifiers [J]. *Neural Processing Letter*, 1999, 9(3): 293-300.

[4] 阎威武, 邵惠鹤. 支持向量机和最小二乘支持向量机的比较及应用研究[J]. *控制与决策*, 2003, 18(3): 358-360.
(Yan W W, Shao H H. Application of support vector machines and least squares support vector machines to heart disease diagnoses[J]. *Control and Decision*, 2003, 18(3): 358-360.)

[5] Lin C J. Formulations of support vector machines: A note from an optimization point of view [J]. *Neural Computation*, 2001, 13(2): 307-317.

[6] Zhang X G. Using class-center vectors to build support vector machines [A]. *Neural Networks for Signal Processing IX - Proc of the* 1999 *IEEE Workshop* [C]. Wisconsin: IEEE Inc, 1999: 33-37.

[7] Lin C F, Wang S D. Fuzzy support vector machines [J]. *IEEE Trans on Neural Networks*, 2002, 13(3): 466-471.

[8] Huang H P, Liu Y H. Fuzzy support vector machines for pattern recognition and data mining [J]. *Int J of Fuzzy Systems*, 2002, 4(3): 3-12.

[9] Tax D M J, Duin R P W. Data domain description by support vectors[A]. *Proc of 8th European Symposium on Artificial Neural Networks*[C]. Brussels: Facto D, 1999: 251-256.

(上接第 620 页)

[6] 郑有才, 蔡希尧. 元数据驱动的可通用通信软件的设计[J]. *西安电子科技大学学报*, 1998, 25(6): 778-781.
(Zhen Y C, Cai X Y. The implementation of metadata-driven reusable communication software [J]. *J of Xidian University*, 1998, 25(6): 778-781.)

[7] Bharat Jayaraman. Semantics of EqL [J]. *IEEE Trans on Software Engineering*, 1988, 14(4): 472-480.

[8] Kewley Robert Hargreaves Jr. Computational intelligence for support of military tactical decision making [D]. Troy: Rensselaer Polytechnic Institute, 2000.

[9] Amit Jayant Patel. Obstal: A language with objects, subtyping, and classes [D]. Stamford: Stanford: University, 2001.

[10] Subrahmanyam P A, Singh K J, Guy Story, et al. Quality assurance in scripting[J]. *IEEE Multimedia*, 1995, 2(2): 50-59.

[11] Jun Miura, Motokuni Ito, Yoshiaki Shirai. A three-level control architecture for autonomous vehicle driving in a dynamic and uncertain traffic environment [A]. *Proc of IEEE Conf on Intelligent Transportation Systems* [C]. Boston: IEEE Press, 1997: 706-711.