第 34 卷 第 4 期 计 算 机 工 程 2008 年 2 月
Vol.34 No.4 Computer Engineering February 2008
·软件技术与数据库· 文章编号：1000—3428(2008)04—0096—02 文献标识码：A 中图分类号：TP319

一种规则驱动的觉察上下文计算模式

何秋生，涂时亮
(复旦大学计算机科学与工程系，上海 200433)

摘 要：以面向服务的体系结构和产生式推理技术为基础，提出一种规则驱动的自适应觉察上下文计算模式。在该模式中，用户的需求及服务复合策略的语义用产生式表示。在动态普适计算运行环境中，规则推理引擎能够在运行期间，根据当前的上下文知识自动选择服务构件，动态合成应用软件完成预定目标。

关键词：觉察上下文计算；面向服务的体系结构；产生式；自适应

Rule-driven Context-aware Computing Model

HE Qiu-sheng, TU Shi-liang
(Department of Computer Science and Engineering, Fudan University, Shanghai 200433)

【Abstract】 Leveraging Service-Oriented Architecture(SOA) and reasoning techniques based on production, this paper proposes a rule-driven adaptive context-aware computing model. Production is used as semantic basis to model and represent user's requirements and service composition policies. During the runtime, the reasoning engine automatically specifies service components for the satisfaction of preloaded goals depending on changing availability of resources in dynamic ubiquitous environments.

【Key words】 context-aware computing; Service-Oriented Architecture(SOA); production; adaptation

觉察上下文计算(context-aware computing)[1]是以人为中心的普适计算的基础之一，其关键为应用软件基于上下文的自适应性[2]。在高度动态化的复杂普适计算环境下，自适应性应由软件基础设施保障，使上下文对应用软件透明，从而降低应用软件开发的复杂性。

1 系统的层次模型

本文提出了一种规则驱动的觉察上下文的计算模式，基于面向服务的开放式Java平台OSGI(Open Service Gateway Initiative)[3]，构建轻量级的、面向服务的、觉察上下文的计算平台LCASOA(Lightweight Context-Aware Service-Oriented Architecture)。遵循策略-机制、接口-实现分离的原则，自上而下LCASOA可分为和环境逐次紧密的 4 个层次：目标层(goal)，策略层(policy)，概念服务层(service)，执行层(implementation)，各层均可按需访问环境上下文见图 1。

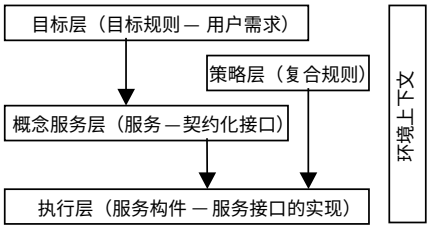

图 1 LCASOA 系统的层次模型

LCASOA 中存在两类规则：目标规则表示用户对服务质量和功能需求的抽象，由用户提供；复合规则表示服务复合策略的抽象，例如服务依赖关系，由对运行环境深入了解的管理员或系统设计者提供。两类规则均作用于以主体-属性-值对描述的环境上下文上。规则所构成的知识库面向特定的领域，以限定规则库规模，使得 LCASOA 能够在资源受限制的嵌入式设备中运行。

在 LCASOA 中，第 3 层的概念服务描述特定领域中各类计算实体承诺的功能，即契约化接口。概念服务由运行环境中所部署的属于执行层的服务构件提供。一项服务可能同时存在多个潜在的提供者，在运行期间，LCASOA 的核心模块——规则推理引擎能够根据目标和复合规则，以及即时的上下文进行演算推理，自动选择合适的服务构件作为基本构建块动态复合为应用程序，以满足用户需求。

LCASOA 具有以下特点：

(1)通过 4 个层次的抽象，应用软件开发者可摆脱需要过分关注目标运行环境细节的困境。

(2)采用 SOA，以松耦合的服务作为应用软件的动态构建块，使得应用软件可以在运行期间调整其行为，以自动适应普适计算环境及用户需求的动态变化。

(3)应用软件的自适应性的基础有 3 个：觉察上下文的计算范式，主动性的产生式系统，及两者决定的 SOA 服务复合和绑定过程。因为上下文知识库和规则知识库在具体运行环境下是确定的，所以服务复合和绑定的结果是可预测的，即基于 LCASOA 的系统是可预测的自治系统。

2 上下文模式

一个好的觉察上下文计算系统必然有一个合理的上下文模式。一般上下文信息可分为高层和低层两类，前者指时间、地点、网络连接速率等物理世界的直观属性，后者指用户兴趣、服务质量要求等抽象信息，高层信息往往由低层信息复合产生。常见的上下文模式有键-值、图形、面向对象、本体、逻辑等[4]。

由于 LCASOA 面向小型设备，因此对于高层和低层两类

作者简介：何秋生(1973－)，男，博士研究生，主研方向：普适计算，面向服务的体系结构；涂时亮，教授、博士生导师

收稿日期：2007-04-25 **E-mail：**heqiusheng@fudan.edu.cn

上下文，本文均采用简单而高效的主体-属性-值模式表示，该模式定义如下：

定义 1 特定领域的上下文信息可用三元组{主体，属性，值}表示，其中，主体 $subject \in S^*$，S 表示特定领域参与者的集合，包括人物、具有独立功能的计算实体等；属性 $attribution \in A^*$，A 表示主体属性的集合，A 依赖于 S；值 $value \in V^*$；V 表示 A 中元素取值的集合。

依据此定义，本文提出了上下文模式语言 CPL(Context Modeling Language)用于描述上下文信息。CPL 采用前缀格式，其语法的简化 BNF 描述如下：

```
CPL 表达式::= 复杂项 | 复合项 | 单项
复杂项::= 复合项 (复合项+)
复合项::= (逻辑操作 (单项 单项+))
逻辑操作::= 非 | 与 | 或
单项::= (关系 主体.属性 值) 时间戳
关系::= > | < | = | >= | <= | ~=
主体::= 标记
属性::= 标记
值::= 标记 | 数字| 数字..数字
时间戳::= 整形
标记::= 字母(字母 | 数字 | '_')*
数字::= 整形 | 浮点
整形::= [0..9]+
浮点::= [1-9]+.[1-9]*
字母::= [a..zA..Z]
```

从 CPL 语法可知，一条 CPL 表达式可完整表达多条满足定义 1 的上下文信息或取值结果。CPL 表达式包含的时间戳表示上下文信息产生的时间，这为解决上下文信息在时间域中的冲突提供了一种简单而有效的手段。此外，CPL 语法和 LDAP(Lightweight Directory Access Protocol)所采用的语法相似，原因是 OSGI 采用了类 LDAP 的过滤器用于发现服务提供者的查找过程。因此，作为产生式规则条件的 CPL 表达式在产生式触发时可以直接作用于服务复合和绑定过程，方便快捷。除了作为产生式规则条件，LCSOA 中上下文信息存储、访问也是基于 CPL 的。

3 规则及推理系统

3.1 产生式

本文采用 IF-THEN 形式的产生式表示本模式涉及的 2 类规则：目标规则和复合规则。目标规则来自于 LCASOA 平台之外的规划器(planner)，它预处理以 Prolog 描述的高层次的用户需求，得到 LCASOA 的输入——规划。一条目标规则表示规划的一个原子目标(atomic goal)。规划器的预处理过程较复杂，因此，一般在资源丰富的计算设备上完成。

LCASOA 产生式基于 Java 语言，可称之为 PJS(Production based on Java Syntax)。PJS 直观易读，便于用户理解使用，其语法的简化 BNF 描述如下所示：

```
规则::= "Goal:" | "Comp:" 标识 规则体 "End"
规则体::= 声明 条件 动作
声明::= " Objects:" (类名 标识 ("," 标识)* ";")*
条件::= "If:" (CPL 表达式)*
动作::= "Then:" (表达式)*
```

其中，条件部分(即模式)使用 CPL 表达式描述；动作部分的表达式指一般的 Java 语句，一旦某个产生式被激活，这些 Java 语句将被执行；声明部分声明 Java 语句所使用的对象。

产生式有两种办法进入知识库：利用 LCASOA 提供的编程 API，使用在运行期载入并解析的脚本。前者效率高，后者方便。

3.2 复合规则

和目标规则不同，服务复合产生式的条件除了上下文信息之外，还牵涉其他信息，其中，最重要的是定义 2 所示的服务之间内在的依赖关系。

定义 2 有服务 u,v，若 u 需要至少一种 v 提供的功能，例如调用 v 的一个方法，则称 u 依赖于 v，记为 $u \xrightarrow{\ominus} v(u \rightarrow v)$。记号$\ominus$表示依赖模式，可用一个四元组$(T, C, A, F)$表示，其中，$T$ 表示依赖类型，有静态和动态两种，静态运行期内绑定不可改变；C 表示映射模式(cardinality)，例如 $0..n$ 指表示 u 可以绑定到任意多个 v 的实例，可能的多重性有 $0..n$,$0..1$, $1..n$,$1..1$,$n..n$ 等；A 表示访问模式，即 u 依赖于 v 的多个实例时，如何选择这些实例，具体模式有全部选择、基于 LDAP 语法过滤器、基于优先级、最先匹配、round-robin、随机等；F 表示 u 需要 v 提供的功能。

多个服务之间依赖关系可用有向图表示。在普适环境中，服务集合和服务依赖图呈现动态和复杂的特点，因此，表达其语义的复合规则需要领域专家主导设计。

3.3 推理系统

推理系统使用一种Rete[5]变种算法，采用首序及前向链逻辑进行推理。在运行期的推理引擎的活动大致可分为以下 4 个步骤：

(1)上下文感知服务不断搜集周围环境的信息，并按需通知推理引擎；

(2)推理引擎在目标规则集合上进行推理，确定当前须满足的目标集合；

(3)推理引擎根据服务复合规则，建立服务依赖图，确定当前的服务集合；

(4)通过 SOA 服务注册器(registrar)，查找匹配的服务提供者——构件，完成服务绑定，即完成应用软件的动态复合。服务构件激活运行，达成用户目标。

在上述过程中，步骤(2)、步骤(3)可以并行，也可以统一在一个推理过程中完成。同时，LCASOA 提供 LRU(Least Recently Used)、优先级等策略以解决激活规则冲突的问题。以上 4 个步骤在系统生命周期内将被反复执行。

4 结束语

本文结合面向服务的体系结构、产生式系统和觉察上下文计算范式，提出了一个规则驱动的觉察上下文计算模式，设计了 CPL 描述上下文信息，以及 PSJ 表达用户需求和服务自动复合策略的语义，在此基础上，建立了面向服务的觉察上下文平台 LCASOA。LCASOA 具有自适应、动态、开销小的特点，适用于普适计算环境中广泛部署的小型的嵌入式设备，也为 SOA 系统引入了可预测的服务复合机制。

下一步工作将进一步完善 LCASOA，并研究基于分布式环境的远程服务复合的机制。

参考文献

[1] Schilit B, Adams N, Want R. Context-aware Computing Applications[C]//Proc. of IEEE Workshop Mobile Computing Systems and Applications. Los Alamitos, Calif.: [s. n.], 1994: 85-90.

[2] 徐光祐, 史元春, 谢伟凯. 普适计算[J]. 计算机学报, 2003, 26(9): 1042-1050.

(下转第 100 页)