arXiv:2303.00794v2 [cs.LO] 6 Mar 2023

# 由命题配置逻辑描述的基于组件的系统的动态重配置

George Rahonis[a], 和Melpomeni Soula[b]塞萨洛尼基亚里士多德大学数学系 54124 塞萨洛尼基，希腊[a]grahonis@math.auth.gr，[b] soulmelp@math.auth.gr

摘要

我们研究架构由命题配置逻辑公式描述的动态可重构基于组件的系统。我们基于众所周知的架构给出了若干可重构系统的示例，并陈述了初步的可判定性结果。

关键词：动态可重构的基于组件的系统；架构；命题配置逻辑。

## 1 引言

系统工程中最严谨的形式体系之一是所谓的基于组件的方法。基于组件的形式体系为复杂系统提供严格的数学表示，并支持规约语言（例如参见 [24, 31]），还支持对（动态）重配置等重要特征进行严格描述 [19]。基于组件的系统由有限种不同类型的组件（通常表示为迁移标记系统）组成，每种组件还可有若干副本，称为实例。组件实例之间通过各自的接口（即端口）进行通信。具体而言，不同组件的端口通过交互相互连接。交互集合遵循特定规则，这些规则决定组件系统的拓扑结构和运行方式。规则集合由架构描述。表示架构的方式有多种；逻辑已被证明非常可靠，而且能够得出重要结论（例如参见[24, 29, 31]）。另一方面，许多实际应用（如物联网系统）采用基于组件的系统进行描述，其中某些组件的实例数量会动态变化，即可以连接一些新实例，也可以停用另一些实例，等等。然而，在动态重配置过程中，可能会发生故障交互，进而影响系统底层架构的正确实现。如何判断基于组件的系统进行动态重配置后是否仍保留其主要特征，以及若未能保留，是否有可能对其进行“纠正”，是一项挑战。

本文提出了一种仍在研究中的方法，该方法基于命题配置逻辑（简称 PCL）。PCL 在 [24], 中被引入，为描述基于组件的系统架构提供了一种严格的数学规约语言。几乎所有

已知架构都可以用 PCL 及其一阶和二阶层级中的公式表示。PCL 也已扩展至定量化框架（参见 [21, 29]）。在我们的模型中，基于组件的系统的重配置，以及由此产生的架构实现的重配置，由一种赋值映射 $\sigma$ 描述，该映射会激活或停用组件实例之间的交互。我们证明，可以判定这种无限赋值是否能确保架构最终得到正确实现（即经过无限长的“时间”之后）。若不能，我们将探究是否可以通过限制 $\sigma$ 来获得一种“正确”的赋值。

## 2 相关工作

目前已有多种方法用于描述基于组件的系统的重配置。在[2, 5, 23, 25, 32] 中，作者通过构建若干软件工具研究了动态可重构架构问题；而在 [22] 和 [12] 中，则分别使用 $\pi$-演算和 $\pi$-ADL研究架构的重配置。人们还使用了多种图文法模型来研究架构的动态变化 [9, 10, 20]。在 [8] 中，资源逻辑被用作支持参数化架构和可重构架构局部推理的工具。在 [15]中，构建了一个基于线性时序逻辑、用于面向服务的可重构架构的工具。

我们的工作与近期论文 [19]的关系更为密切。在该论文中，作者引入了一种时序配置逻辑，即一种原子命题为配置逻辑公式的线性时序逻辑，用于描述可重构的基于组件的系统的架构。数量未知的组件实例由PCL的一阶和二阶层次表示。此外，借助这两个层次，作者能够描述环形结构等架构；众所周知，这类架构无法用PCL中的公式描述。在我们的理论中，我们假设每种组件类型的实例数量都是有界的。“界”可以任意大，例如可以涵盖现实世界中应当存在的所有组件实例。然后，这些数量可以直接展开到我们的逻辑公式中。我们证明，大多数众所周知的架构都可以用我们的逻辑中的公式描述，尽管需要配置逻辑一阶和二阶层次的架构（如环形结构、网格等）目前仍不在我们的研究范围之内。我们的逻辑的优势在于所得到的可判定性结果。

## 3 预备知识

对于每个自然数 $n > 0$，我们定义 $[n] = \{1, \ldots, n\}$。令 $A$ 为一个字母表，即一个非空有限集。通常，A 上所有无限字的集合记作 $A^\omega$。一个（非确定性）B ¨uchi 自动机是一个五元组 $\mathcal{A} = (Q, A, I, \Delta, F)$，其中 $Q$ 是一个有限状态集，$A$ 是输入字母表，$I \subseteq Q$ 是初始状态集，$\Delta \subseteq Q \times A \times Q$ 是转移集，$F \subseteq Q$ 是终态集。

设 $w = a_0 a_1 ... \in A^\omega$. A $\mathcal{A}$ 上的路径是一个无限转移序列 $P_w = \left((q_i, a_i, q_{a_{i+1}})\right)_{i \geq 0}$，其中对每个 $i \geq 0.$，都有 $(q_i, a_i, q_{a_{i+1}}) \in \Delta$。我们用 $In^Q(P_w)$ 表示沿着 $P_w$无限多次出现的状态集合。那么，如果 $P_w\,(P_w) \cap F \neq \emptyset$，则称 $P_w$ 是成功的。无限字 $w \in A^\omega$ 被 $\mathcal{A}$ 识别（或接受），当且仅当存在 $\mathcal{A}$上的一条成功路径 $w.$。$\mathcal{A}$ 的语言（或行为），记作 $L(\mathcal{A})$，是所有被 $\mathcal{A}$接受的字的集合。

接下来，我们回顾命题交互逻辑和配置逻辑（参见 [24]）。为此，我们考虑一个非空有限集 $P$，其元素称为端口。令 $I(P) = \mathcal{P}(P) \setminus \{\emptyset\}$，其中 $\mathcal{P}(P)$ 表示 $P$的幂集。 $I(P)$的元素称为交互。于是，命题交互逻辑（简称 PIL）公式在 $P$ 上的语法由文法 $\phi ::= \mathsf{true} \mid p \mid \overline{\phi} \mid \phi \vee \phi$ 给出，其中 $p \in P$。对于每个 PIL 公式 $\phi$，我们定义 $\overline{\overline{\phi}} = \phi$，并且 $\mathsf{false} = \mathsf{true}$。两个 PIL 公式 $\phi$、$\phi'$ 的合取定义为 $\phi \wedge \phi' := \overline{(\overline{\phi} \vee \overline{\phi'})}$。形如 $p_1 \wedge \ldots \wedge p_n$ 且满足 $n > 0$，以及形如 $p_i \in P$ 或 $p_i = \overline{p'_i}$ 且满足对每个$1 \leq i \leq n$都有 $p'_i \in P$的 PIL 公式，称为单项式。我们用 $p_1 \ldots p_n$表示单项式 $p_1 \wedge \ldots \wedge p_n$。设 $\phi$ 是一个 PIL 公式， $a \in I(P)$ 是一个交互。我们通过对 $\phi$的结构进行归纳，定义满足关系 $a \models_{\mathrm{PIL}} \phi$如下：

- $a \models_{\mathrm{PIL}}$ 真，
- $a \models_{\mathrm{PIL}} p$ 当且仅当 $p \in a$，
- $a \models_{\mathrm{PIL}} \overline{\phi}$ 当且仅当 $a \not\models_{\mathrm{PIL}} \phi$，
- $a \models_{\mathrm{PIL}} \phi_1 \vee \phi_2$ 当且仅当 $a \models_{\mathrm{PIL}} \phi_1$ 或 $a \models_{\mathrm{PIL}} \phi_2$.

显然，对于每个 $a \in I(P)$，都有 $a \not\models_{\mathrm{PIL}}$ 假。对于每个交互 $a$，我们定义其特征单项式 $m_a = \bigwedge_{p \in a} p \wedge \bigwedge_{p \notin a} \overline{p}$。于是，对于每个 $a' \in I(P)$，显然有 $a' \models_{\mathrm{PIL}} m_a$ 当且仅当 $a' = a$。

在 $P$ 上的命题配置逻辑（简称 PCL）公式的语法由以下语法规则给出： $f ::= \mathsf{真} \mid \phi \mid \neg f \mid f \sqcup f \mid f + f$其中 $\phi$表示一个在 $P$ 上的 PIL 公式。算子 $\neg$、 $\sqcup$和 $+$ 分别称为补、并和合并。交算子 $\sqcap$定义如下： $f_1 \sqcap f_2 := \neg(\neg f_1 \sqcup \neg f_2)$。我们令 $C(P) = \mathcal{P}(I(P)) \setminus \{\emptyset\}$。对于每个 PCL 公式 $f$ 以及 $\gamma \in C(P)$，我们按以下方式根据 $f$ 的结构归纳定义满足关系 $\gamma \models f$：

- $\gamma \models \mathrm{true}$,
- $\gamma \models \phi$ iff $a \models_{\mathrm{PIL}} \phi$ for every $a \in \gamma$,
- $\gamma \models \neg f$ iff $\gamma \not\models f$,
- $\gamma \models f_1 \sqcup f_2$ iff $\gamma \models f_1$ or $\gamma \models f_2$,
- $\gamma \models f_1 + f_2$ iff there exist $\gamma_1, \gamma_2 \in C(P)$ such that $\gamma = \gamma_1 \cup \gamma_2$, and $\gamma_1 \models f_1$ and $\gamma_2 \models f_2$.

于是，我们显然得到
- $\gamma \models f_1 \sqcap f_2$ iff $\gamma \models f_1$ and $\gamma \models f_2$, and
- $\gamma \models f_1 \implies f_2$ iff $\gamma \not\models f_1$ or $\gamma \models f_2$.

我们将每个 PCL 公式 $f$ 的闭包 $\sim f$ 定义为
- $\sim f := f +$真,

并将两个 PCL 公式 $f_1$ 和 $f_2$ 的析取 $f_1 \vee f_2$ 定义为
- $f_1 \vee f_2 := f_1 \sqcup f_2 \sqcup (f_1 + f_2)$。

若对于每个 $\gamma \in C$（$P$）， $\gamma \models f$ 当且仅当 $\gamma \models f'$，则称两个 PCL 公式 $f$、 $f'$ 等价，记作 $f \equiv f'$。

由于对于每个交互公式 $\phi$， $\phi \wedge \phi' \equiv \phi \sqcap \phi'$，因此下文中我们用相同的符号 $\wedge$ 表示 PCL 公式的合取和交集运算。

如果 PCL 公式 $f$ 满足以下条件，则称其为完全范式:

$$f = \bigsqcup_{i \in I} \sum_{j \in J_i} m_{i,j}$$

其中，对于每个 $i \in I$，指标集 $I$ 和 $J_i$都是有限的，且 $m_{i,j}$ 均为完全单项式，即形式为 $\bigwedge_{p \in P_+} p \wedge \bigwedge_{p \in P_} \overline{p}$ 的单项式，其中 $P_+ \cup P_ = P$和 $P_+ \cap P_ = \emptyset$。

# 4 架构

我们将介绍基于组件的系统及其架构。一个基于组件的系统由有限个相同类型或不同类型的组件构成。在许多知名的基于组件的系统中，包括 BIP [7], 、REO [3], 、X-MAN [18], 和 B [1] ，组件均定义为标记迁移系统（简称 LTS）。本文采用相同的定义，并沿用 BIP框架中基本概念的术语。组件之间的通信通过各自对应的接口实现，即通过相关联的标签集合实现，这些标签称为端口。更准确地说，组件之间的通信由交互定义，即由端口集合定义。

原子构件是一个 LTS $B = (Q, P, q_0, R)$，其中 $Q$ 是状态的有限集， $P$ 是端口的有限集， $q_0$ 是初始状态， $R \subseteq Q \times P \times Q$ 是转移集。

当涉及多个原子构件时，原子构件 $B$ 将称为组件。此外，对于每个组件集 $\mathcal{B} = \{B(i) \mid i \in [n]\}$ ，其中 $B(i) = (Q(i)\ P(i)\ q_0(i)\ R(i))$， $i \in [n]$，我们假设对于每个$1 \leq i \neq i' \leq n$，$(Q(i) \cup P(i)) \cap (Q(i') \cup P(i')) = \emptyset$。

设 $\mathcal{B} = \{B(i) \mid i \in [n]\}$ 为一个组件集。令 $P_{\mathcal{B}} = \bigcup_{i \in [n]} P(i)$ 表示 $\mathcal{B}$ 中各元素的所有端口的集合。 $\mathcal{B}$ 的交互是满足以下条件的交互 $a \in I(P_{\mathcal{B}})$：对于每个 $i \in [n]$，$|a \cap P(i)| \leq 1$。如果 $p \in a$，则 $p$ 在 $a$ 中处于激活状态。我们用 $I_{\mathcal{B}}$ 表示 $\mathcal{B}$ 的所有交互的集合，即

$$I_{\mathcal{B}} = \{a \in I(P_{\mathcal{B}}) \mid |a \cap P(i)| \leq 1 \text{ for every } i \in [n]\}$$

并令

$$C_{\mathcal{B}} = \mathcal{P}(I_{\mathcal{B}}) \setminus \{\emptyset\}.$$

定义 1 架构是一个二元组（$\mathcal{B}, f$），其中 $\mathcal{B} = \{B(i) \mid i \in [n]\}$ 是一个组件集合，且对于每个 $i \in [n]$， $B(i) = (Q(i)\ P(i)\ q_0(i)\ R(i))$，而 $f$是关于$P_{\mathcal{B}}$的 PCL 公式。

为了定义可重构架构，我们需要扩展基于组件的系统的概念[1]。具体而言，设 $\mathcal{B} = \{B(i) \mid i \in [n]\}$ 为组件类型集合。对于每个 $i \in [n]$ ，我们取一个自然数 $u_i > 0$ ；对于每个 $j \in [u_i]$ ，取一个副本 $B(i,j) = (Q(i,j)\ P(i,j)\ q_0(i,j)\ R(i,j))$，它属于 $B(i)$，称为 $B(i)$ 的第 $j$ 个实例。因此，对于每个 $i \in [n]$ 和 $j \in [u_i]$，实例 $B(i,j)$ 也是一个组件，我们称其为组件

[1]事实上，我们使用了 [31] 的参数化组件系统这一概念的一个略作修改的版本

实例。我们规定，当且仅当 $i \neq i'$ 或 $j \neq j'$ 时，$(Q(i,j) \cup P(i,j)) \cap (Q(i',j') \cup P(i',j')) = \emptyset$，对每个 $i$、$i' \in [n]$ 和 $j$、$j' \in \bigcup_{i \in [n]} [u_i]$ 均成立。此限制是区分不同组件实例所必需的。它还使我们能够毫无混淆地使用记号 $P(i,j) = \{p(j) \mid p \in P(i)\}$，对每个 $i \in [n]$ 和 $j \in [u_i]$ 均成立。我们用 $p\mathcal{B}$ 表示导出的基于组件的系统，即 $p\mathcal{B} = \{B(i,j) \mid i \in [n], j \in [u_i]\}$。$p\mathcal{B}$ 的端口集合定义为 $P_{p\mathcal{B}} = \bigcup_{i \in [n], j \in [u_i]} P(i,j)$。其交互集合定义为

$$I_{p\mathcal{B}} = \{a \in I(P_{p\mathcal{B}}) \mid |a \cap P(i,j)| \leq 1 \text{ for every } i \in [n], j \in [u_i]\}$$

并令

$$C_{p\mathcal{B}} = \mathcal{P}(I_{p\mathcal{B}}) \setminus \{\emptyset\}.$$

接下来，令 $p\mathcal{B} = \{B(i,j) \mid i \in [n], j \in [u_i]\}$ 为一个基于组件的系统。在 $p\mathcal{B}$ 上的一个实现$\sigma$ 是一个映射序列 $\sigma = (\sigma_l)_{l \geq 0}$，其中 $\sigma_l$：$I_{p\mathcal{B}} \to$ {激活,未激活}。对于每个$k \geq 0$ 和 $\gamma \in C_{p\mathcal{B}}$，我们令 $\sigma_k(\gamma) =$ 激活，当且仅当对于每个 $a \in \gamma$，$\sigma_k(a) =$ 激活；否则，我们令 $\sigma_k(\gamma) =$未激活。

定义 2 动态可重构架构（简称 dra）是一个三元组 $(p\mathcal{B}, f, \sigma)$，其中 $p\mathcal{B} = \{B(i,j) \mid i \in [n], j \in [u_i]\}$ 是一个基于组件的系统，$f$是关于$P_{p\mathcal{B}}$的 PCL 公式，而 $\sigma$ 是在 $p\mathcal{B}$上的一个实现。

对于每个 $l \geq 0$，我们令

$$\gamma(\sigma_l) = \{a \in I_{p\mathcal{B}} \mid \sigma_l(a) = \text{active}\}$$

并将 $f$ 被 $\sigma_l$ 满足定义如下:

$$\sigma_l \models f \quad \text{iff} \quad \gamma(\sigma_l) \models f.$$

然后，如果存在一个自然数 $g \geq 0$，使得对于每个 $l > g$ 都有 $\sigma_l \models f$，我们就称实现 $\sigma$ 对于 $f$ 是可信的。$\sigma$ 可信的含义是，在经过一段“时间” $g$ 后，对于每个 $l > g$，由 $\sigma_l$ 确定的激活交互集合都满足公式 $f$，因此由 $f$ 描述的架构始终得到了正确实现。

备注 3 根据上述构造，定义在 $p\mathcal{B}$ 上的每个实现 $\sigma = (\sigma_l)_{l \geq 0}$ 都定义了一个序列 $\gamma = (\gamma(\sigma_l))_{l \geq 0}$，其中包含 $C_{p\mathcal{B}}$ 中的交互集合。反过来，$C_{p\mathcal{B}}$ 中元素的每个序列 $\gamma = (\gamma_l)_{l \geq 0}$ 都确定了定义在 $p\mathcal{B}$ 上的一个实现 $\sigma(\gamma) = (\sigma(\gamma)_l)_{l \geq 0}$，其中对于每个 $a \in I_{p\mathcal{B}}$，若 $a \in \gamma_l$，则 $\sigma(\gamma)_l(a)$ 激活；否则 $\sigma(\gamma)_l(a) =$ 未激活。因此，定义在 $p\mathcal{B}$ 上的实现与 $C_{p\mathcal{B}}$ 中的交互集合序列一一对应。

### 4.1 示例

接下来，我们将介绍若干适用于包含任意数量组件的基于组件的系统的知名架构的 PCL 公式。

示例 4 我们给出一个 PCL 公式，用于描述被视为动态可重构的主从架构。该架构由两种组件类型构成，即主组件，记作 $B(1)$，以及从组件，记作 $B(2)$。它们各自都有一个端口，分别记作 $m$ 和 $s$。仅允许主组件与从组件通过类型为 $B$ 的交互建立连接

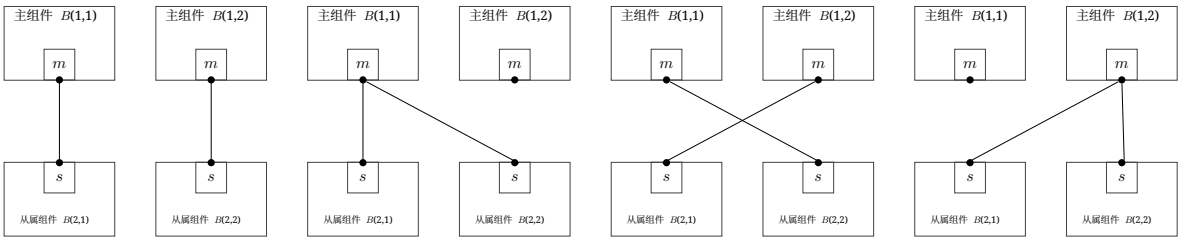


图1：主从架构。

。此外，每个从属实例都可以连接到唯一的主实例。图 1 展示了该架构的一种实例化，其中包含两个主组件和两个从组件，并列出了所有从组件均已连接时允许的交互的全部可能情况。

主实例 $B(1,k)$ 与从属实例$B(2,j)$之间交互的 PIL公式为

$$\phi_{ms}(m(k),s(j)) = m_{\{m(k),s(j)\}}.$$

我们考虑 PCL 公式

$$f = \bigvee_{j\in[u_2]} \left( \bigsqcup_{k\in[u_1]} \phi_{ms}(m(k),s(j)) \right).$$

假设 $\gamma \in C_{p\mathcal{B}}$ 满足 $f$。这意味着至少存在一个从属实例 $B(2,j)$ ($j \in [u_2]$)，它与主实例 $B(1,k)$ ($k \in [u_1]$) 相连，因此该架构处于激活状态。

示例 5 我们为著名的发布-订阅架构（参见 *[4, 6, 14, 28, 30, 34, 35]*）构造一个 PCL 公式。为此，我们定义三种组件类型，即发布者（记作 $B(1)$）、主题（记作 $B(2)$）和订阅者（记作 $B(3)$）。发布者有一个端口，即 $p$；主题有两个端口，即 $t_p$ 和 $t_s$；订阅者有一个端口，即$s$。发布者通过类型为 $\{p,t_p\}$ 的交互与主题相连，订阅者则通过类型为 $\{s,t_s\}$ 的交互与主题相连。

发布者实例 $B(1,k)$ ($k \in [u_1]$) 与主题实例 $B(2,j)$ ($j \in [u_2]$) 之间交互的 PIL公式为

$$\phi_{pt_p}(p(k),t_p(j)) = m_{\{p(k),t_p(j)\}}.$$

订阅者实例 $B(3,r)$ ($r \in [u_3]$) 与主题实例 $B(2,j)$ ($j \in [u_2]$) 之间的交互由以下 PIL公式给出

$$\phi_{st_s}(s(r),t_s(j)) = m_{\{s(r),t_s(j)\}}.$$

接下来，我们令

$$\zeta(t_p(j)) = \bigvee_{k\in[u_1]} \phi_{pt_p}(p(k),t_p(j))$$

*and*

$$\zeta(s(r)) = \bigvee_{j\in[u_2]} (\phi_{st_s}(s(r),t_s(j)) + \zeta(t_p(j))).$$

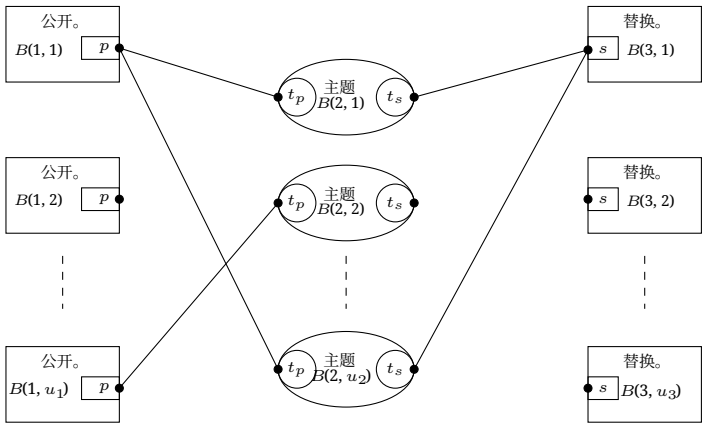


图2：发布-订阅架构。

由此得出 PCL 公式

$$f = \bigvee_{r\in[u_3]} \zeta(s(r)) \vee \bigvee_{j\in[u_2]} \zeta(t_p(j)).$$

不难看出，如果至少有一个订阅者实例 $B(3,r)$ ($r \in [u_3]$) 连接到一个主题实例 $B(2,j)$ ($j \in [u_2]$)，且该主题实例又连接到一个发布者实例 $B(1,k)$ ($k \in [u_1]$)；或者至少有一个主题实例 $B(2,j)$ ($j \in [u_2]$) 连接到一个发布者实例 $B(1,k)$ ($k \in [u_1]$)，那么相应的交互集合 $\gamma \in C_{p\mathcal{B}}$ 满足 PCL 公式 $f$。图2展示了一个包含 $u_1$ 个发布者、 $u_2$ 个主题和 $u_3$ 个订阅者的动态可重构系统快照，其中激活的交互满足公式 $f$。

示例6 我们研究管道-过滤器架构。它包含两种类型的组件：管道，用作待转换数据流的连接器；以及过滤器，对数据执行转换并处理所接收的输入，分别表示为 $B(1)$ 和 $B(2)$。两者各有两个端口，分别为 $in_p$、 $out_p$ 和 $in_f$、 $out_f$。每个过滤器都通过类型为 $\{in_f, out_p\}$ 和 $\{out_f, in_p\}$ 的交互，分别连接到两个不同且唯一的管道。此外，每个管道最多可通过交互 $\{in_f, out_p\}$ 连接到一个过滤器，并且至少有两个特定的已连接管道实例作为管道流水线的两端，即第一个组件的 $in_p$ 和最后一个组件的 $out_p$ 不与其他实例相连。

过滤器实例 $B(2,j)$ ($j \in [u_2]$) 与管道实例 $B(1,k)$ ($k \in [u_1]$) 之间类型为 $\{in_f, out_p\}$ 的交互由以下 PIL公式给出

$$\phi_{in_f out_p}(in_f(j), out_p(k)) = m_{\{in_f(j), out_p(k)\}}.$$

相应地，过滤器实例 $B(2,j)$（$j \in [u_2]$）与管道实例 $B(1,k)$（$k \in [u_1]$）之间类型为 $\{out_f, in_p\}$ 的交互由 PIL公式表示为

$$\phi_{out_f in_p}(out_f(j), in_p(k)) = m_{\{out_f(j), in_p(k)\}}.$$

现在，我们将构建描述管道-过滤器架构的 PCL 公式 $f$。首先，我们考虑 PCL 公式

$$\zeta_1(in_f(j), out_f(j)) = \bigsqcup_{\substack{i_1 \in [u_1] \\ i_2 \in [u_1] \setminus \{i_1\}}} \left(\phi_{in_f out_p}(in_f(j), out_p(i_1)) + \phi_{out_f in_p}(out_f(j), in_p(i_2)) \wedge \xi\right),$$

其中

$$\xi = \bigwedge_{i_1' \in [u_1] \setminus \{i_1\}} \neg\phi_{in_f out_p}\left(in_f(j), out_p(i_1')\right) \wedge \bigwedge_{i_2' \in [u_1] \setminus \{i_2\}} \neg\phi_{out_f in_p}\left(out_f(j), in_p(i_2')\right).$$

若 $j \in [u_2]$，则公式 $\zeta_1(in_f(j)\ out_f(j))$ 表示存在一些互不相同的索引 $i_1, i_2 \in [u_1]$，使得过滤器实例 $B(2, j)$ 分别通过交互 $\{in_f, out_p\}$ 和 $\{out_f, in_p\}$ 与管道实例 $B(1, i_1)$、$B(1, i_2)$ 相连；此外，子公式 $\xi$ 表示这两条管道是唯一的。接下来，我们令

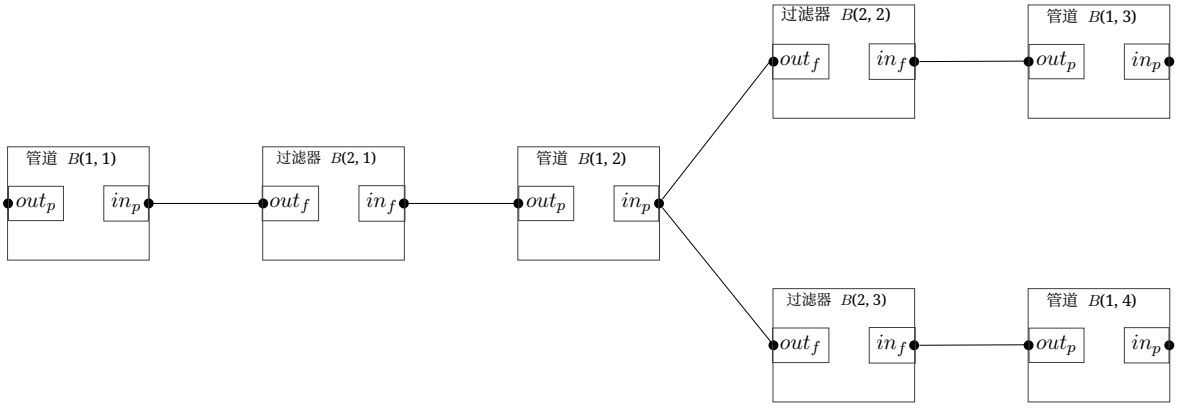


图3：管道-过滤器架构

$$\zeta_2(out_p(r)) = \bigsqcup_{j_1 \in [u_2]} \left( \bigwedge_{j_2 \in [u_2] \setminus \{j_1\}} \neg\phi_{in_f out_p}(in_f(j_2), out_p(r)) \right).$$

如果 $r \in [u_1]$。那么，该公式描述了管道实例 $B(1, r)(r \in [u_1])$ 的输出端口最多可以连接到一个过滤器实例 $B(2, j_1)(j_1 \in [u_2])$。此外，我们定义公式

$$\zeta_3(k_1) = \bigsqcup_{i_1 \in [u_2]} \left(\phi_{in_f out_p}(in_f(i_1), out_p(k_1))\right) \wedge \bigwedge_{i \in [u_2]} \left(\neg\phi_{out_f in_p}(out_f(i), in_p(k_1))\right).$$

$$\zeta_4(k_2) = \bigsqcup_{i_2 \in [u_2]} \left(\phi_{out_f in_p}(out_f(i_2), in_p(k_2))\right) \wedge \bigwedge_{i \in [u_2]} \left(\neg\phi_{in_f out_p}(in_f(i), out_p(k_2))\right).$$

设 $k_1, k_2 \in [u_1]$。公式 $\zeta_3(k_1)$（分别为 $\zeta_4(k_2)$）表示.....的输出端口（分别为输入端口）

管道 $B(1, k_1)$（分别为 $B(1, k_2)$）与过滤器的输入端口（分别为输出端口）交互，但该管道的输入端口（分别为输出端口）未连接。由此得出 PCL 公式

$$f = \sum_{j\in[u_2]} \zeta_1(in_f(j), out_f(j)) \wedge \bigwedge_{r\in[u_1]} \zeta_2\left(out_p(r)\right) \wedge \bigsqcup_{\substack{k_1\in[u_1]\\ k_2\in[u_1]\setminus\{k_1\}}} \left(\zeta_3\left(k_1\right) \wedge \zeta_4\left(k_2\right)\right).$$

如果 $\gamma \in C_{p\mathcal{B}}$ 满足 $f$，那么显然每个过滤器实例 $B(2, j)$ ($j \in [u_2]$) 都与两个管道实例 $B(1, i_1)$、$B(1, i_2)$ ($i_1, i_2 \in [u_1]$) 相连，因此该架构是激活的。在时间 $g \geq 0$，包含四条管道和三个过滤器的系统实例化如下：其激活交互集合遵循管道-过滤器架构，见图3。

示例7 接下来，我们将为星形架构构造一个 PCL 公式。该架构仅包含一种组件类型 $B(1)$，且该组件类型只有一个端口 $p$。其中一个实例被视为中央组件。其余实例可以与中央组件相连，但必须始终至少有一个实例与中央组件相连，从而使该架构处于激活状态。不允许其他交互。

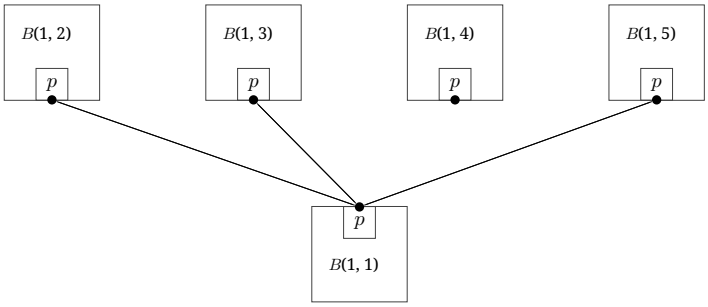


图4：星形架构。

描述实例 $B(1, j)$ 和 $B(1, i)$ 之间交互的 PIL公式（$j, i \in [u_1]$）如下

$$\phi_{pp}(p(j), p(i)) = m_{\{p(j),p(i)\}}.$$

因此，星形架构的 PCL 公式如下

$$f = \bigsqcup_{j\in[u_1]} \bigvee_{i\in[u_1]\setminus\{j\}} \phi_{pp}(p(j), p(i)).$$

如果唯一的中央组件 $B(1, j)$ 与一个独立实例 $B(1, i)$（$i \in [u_1]$）相连，其中 $j \in [u_1]$，那么相应的配置 $\gamma \in C_{p\mathcal{B}}$ 满足 PCL 公式 $f$。

示例8 我们讨论黑板架构 *[11, 13, 26]*，它有助于重要应用（例如参见 *[27, 33]*）。它涉及一个称为黑板的中央组件，记为 $B(1)$，用于呈现待解决问题的状态；独立的知识源组件（$B(2)$）代表为给定问题提供解决方案的专家；以及记为 $B(3)$ 的控制器组件，负责调度

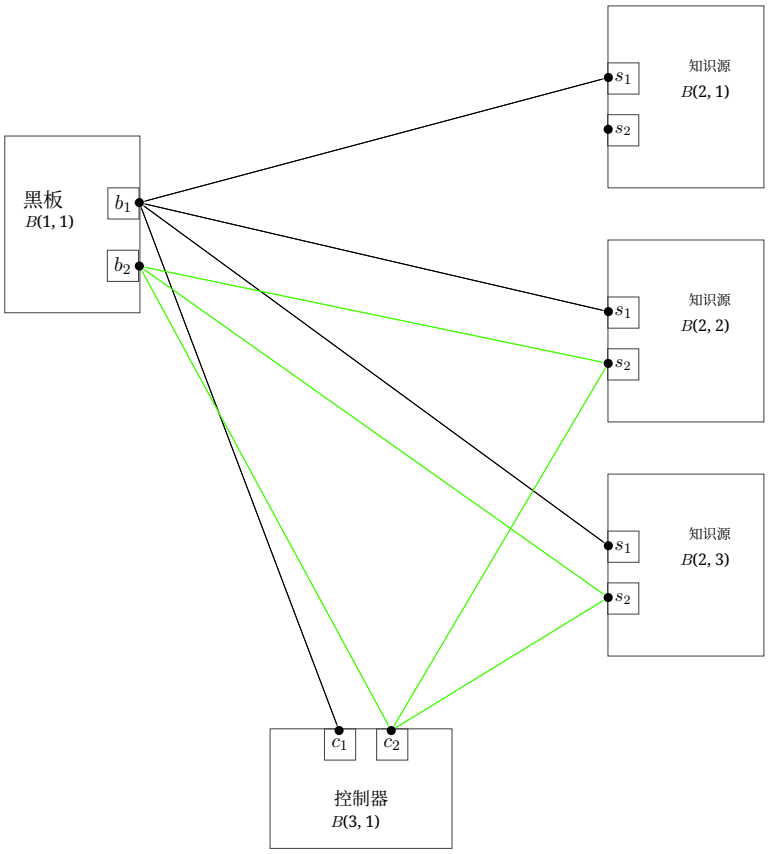


图 5：黑板架构。

候选知识源在黑板上执行其解决方案。每个组件都有两个端口。具体而言，黑板具有端口 $b_1$、$b_2$，知识源具有端口$s_1$、 $s_2$，控制器具有端口 $c_1$ 和 $c_2$。

黑板分别通过 $\{b_1, s_1\}$ 和 $\{b_1, c_1\}$与知识源及控制器交互，以便向知识源呈现问题状态并通知控制器。知识源通过控制器连接到黑板，以便将其解决方案写入黑板；这种连接通过类型为 $\{b_2, s_2, c_2\}$的交互实现。

PIL公式

$$\phi_{b_1 s_1}(b_1(1), s_1(j)) = m_{\{b_1(1), s_1(j)\}}$$

描述了黑板 $B$(1, 1) 与知识源 $B(2, j)$（$j \in [u_2]$）之间的连接，使后者能够针对给定问题从黑板获得信息。黑板 $B$(1,1) 与控制器 $B$(3,1) 之间的连接由下式给出

$$\phi_{b_1 c_1}(b_1(1), c_1(1)) = m_{\{b_1(1), c_1(1)\}}.$$

最后，描述黑板 $B$(1,1)、知识源 $B(2, j)$（$j \in [u_2]$）与控制器 $B$(3,1) 之间交互的 PIL公式为

$$\phi_{b_2 s_2 c_2}(b_2(1), s_2(j), c_2(1)) = m_{\{b_2(1), s_2(j), c_2(1)\}}.$$

描述黑板架构的 PCL 公式为

$$f = \phi_{b_1c_1}(b_1(1), c_1(1)) + \sum_{j\in[u_2]} \phi_{b_1s_1}(b_1(1), s_1(j)) + \bigvee_{j\in[u_2]} \phi_{b_2s_2c_2}(b_2(1), s_2(j), c_2(1)).$$

设 $\gamma \in C_{p\mathcal{B}}$满足 $f$。那么，黑板 $B(1,1)$会向给定问题的所有知识源和控制器发送通知，并且至少有一个知识源连接到黑板和控制器，以便写入其解决方案（关于三个知识源的动态可重构系统快照，见图 5；其中所有激活的交互都满足公式 $f$）。

示例 9 我们继续讨论作为动态可重构系统的请求-响应架构。这种架构在分布式计算中非常流行，适用于各种各样的应用。请求-响应模型包含三种组件类型，即服务、客户端和协调器。我们分别用 $B(1)$、 $B(2)$ 和 $B(3)$ 表示它们。服务包含两个端口 $get_s$ 和 $send$。客户端和协调器各有三个端口，分别为 $con_{cl}$、 $req$、 $rec$ 和 $con_c$、 $get_c$、 $dsc$。服务通过协调器与客户端通信。更确切地说，客户端通过类型为 $\{get_s, req, get_c\}$的交互向服务发送请求。服务处理该请求，并通过类型为 $\{send, rec, dsc\}$的交互返回响应。此外，客户端通过类型为 $\{con_{cl}, con_c\}$的交互连接到协调器。协调器确保每次只有一个客户端连接到某个服务。服务实例不能连接到其他客户端，协调器也不能连接到其他服务。

客户端 $B(2,k)(k \in [u_2])$ 与协调器 $B(3,j)(j \in [u_3])$ 之间的交互由以下 PIL公式给出

$$\phi_{cc}(con_{cl}(k), con_c(j)) = m_{\{con_{cl}(k), con_c(j)\}}.$$

服务 $B(1,i)(i \in [u_1])$、客户端 $B(2,k)(k \in [u_2])$ 和协调器 $B(3,j)(j \in [u_3])$ 之间类型为 $\{get_s, req, get_c\}$ 的交互由以下公式给出

$$\phi_{grg}(get_s(i), req(k), get_c(j)) = m_{\{get_s(i), req(k), get_c(j)\}},$$

类型为 $\{send, rec, dsc\}$ 的交互由以下 PIL公式给出

$$\phi_{srd}(send(i), rec(k), dsc(j)) = m_{\{send(i), rec(k), dsc(j)\}}.$$

我们考虑以下公式

$$\zeta(get_s(i), send(i)) = \bigsqcup_{\substack{k\in[u_2]\\ j\in[u_3]}} \Big(\phi_{cc}(con_{cl}(k), con_c(j)) + \phi_{grg}(get_s(i), req(k), get_c(j)) + \phi_{srd}(send(i), rec(k), dsc(j))\Big)$$

该公式描述了服务实例 $B(1,i)$ 与单个客户端 $B(2,k)$ $(k \in [u_2])$ 通过一个协调器实例 $B(3,j)$ $(j \in [u_3])$ 建立的连接。我们令

$$\xi = \bigwedge_{\substack{i_1\in[u_1],k_1\in[u_2]\\ j_1\in[u_3]}} (\neg\phi_{grg}\,(get_s(i_1), req(k_1), get_c(j_1)) \sqcup (\sim \phi_{grg}(get_s(i_1), req(k_1), get_c(j_1)) \wedge \xi_1))$$

*with*

$$\xi_1 = \bigwedge_{\substack{i_2\in[u_1]\setminus\{i_1\}\\ k_2\in[u_2]}} \neg\phi_{grg}(get_s(i_2), req(k_2), get_c(j_1)).$$

公式 $\xi$ 确保，如果三个实例 $B(1, i_1)$ $B(2, k_1)$ 和 $B(3, j_1)$ 之间存在连接，那么协调器实例 $B(3, j_1)$ 就不能与任何其他服务实例交互。

最后，我们得出 PCL 公式

$$f = \bigvee_{i\in[u_1]} \big(\zeta(get_s(i), send(i)) \wedge \xi\big).$$

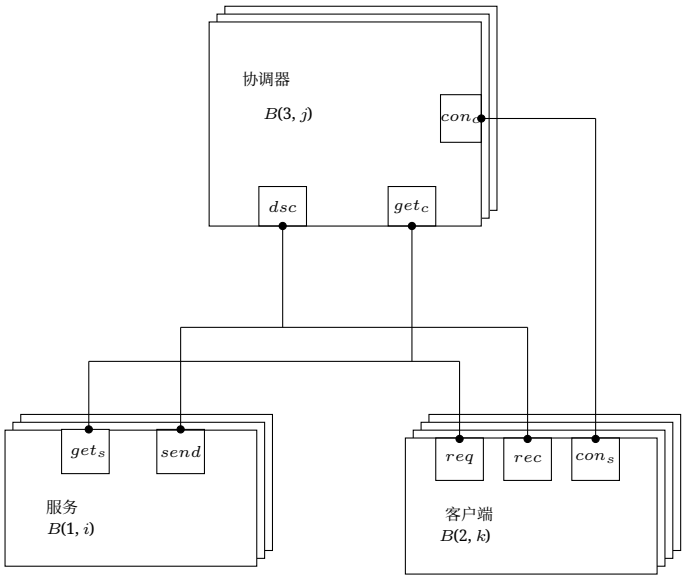


图 6：请求-响应架构。

显然，如果配置 $\gamma \in C_{p\mathcal{B}}$ 满足 $f$，那么至少存在一个服务实例$B(1, i)$ ($i \in [u_1]$)，它通过一个协调器实例 $B(3, j)$ ($j \in [u_3]$) 连接到唯一的客户端 $B(2, k)$ ($k \in [u_2]$)，这意味着该架构处于激活状态。

## 4.2 dra 的可判定性结果

接下来，我们将研究动态可重构架构的可判定性性质。首先，我们证明，对于每个架构 ($p\mathcal{B}$, $f$)，都可以计算出在 $\sigma$ 上对 $f$ 可信的实现集合。

定理 10 设 $(p\mathcal{B}, f)$ 为一个架构。则我们可以在双指数时间内，计算出在 $p\mathcal{B}$ 上对 $f$ 可信的实现集合。

证明。首先，我们用双指数时间计算集合 $C_{p\mathcal{B}}$。接下来，根据定理 4.43，我们可以有效地构造一个与 $f$ 等价的完全范式 PCL 公式 $f'$。这也需要双指数时间。利用 $f'$ 的结构，我们可以在常数时间内确定集合 $C_{p\mathcal{B}}(\models f') \subseteq C_{p\mathcal{B}}$，其中

$$C_{p\mathcal{B}}(\models f') = \{\gamma \in C_{p\mathcal{B}} \mid \gamma \models f'\}.$$

接下来，我们考虑 B¨uchi 自动机 $\mathcal{A}_{f'} = (\{q_0, q_1\}, C_{p\mathcal{B}}, \{q_0\}, \Delta, \{q_1\})$，其

$$\Delta = \{(q_0, \gamma, q_0) \mid \gamma \in C_{p\mathcal{B}}\} \cup \{(q_0, \gamma, q_1) \mid \gamma \in C_{p\mathcal{B}}(\models f')\} \cup \{(q_1, \gamma, q_1) \mid \gamma \in C_{p\mathcal{B}}(\models f')\}.$$

根据 $\mathcal{A}_{f'}$ 的构造可知，序列 $(\gamma_i)_{i\geq 0} \in C^{\omega}_{p\mathcal{B}}$ 被 $\mathcal{A}_{f'}$ 接受，当且仅当存在一个自然数 $g \geq 0$，使得对于每个 $i > g$，都有 $\gamma_i \models f'$，也就是说，对于每个 $i > g$，都有 $\gamma_i \models f$。由备注 3 得出结论。 ■

推论 11 设 $(p\mathcal{B}, f, \sigma)$ 是一个 dra。那么我们可以在双指数时间内判定 $\sigma$ 是否对 $f$ 可信。

证明。根据定理 10，我们可以在双指数时间内计算出 $p\mathcal{B}$ 上对 $f$ 可信的实现集合。因此，我们只需检查 $\sigma$ 是否属于该集合。 ■

接下来，我们定义 dra 的一种等价关系。更确切地说，设 $(p\mathcal{B}, f, \sigma)$ 和 $(p\mathcal{B}', f', \sigma')$ 是两个 dra。若 $p\mathcal{B} = p\mathcal{B}'$、$f \equiv f'$ 和 $\sigma$、$\sigma'$ 分别对 $f$ 和 $f'$ 可信，则称 $(p\mathcal{B}, f, \sigma)$ 和 $(p\mathcal{B}', f', \sigma')$ 等价。下面给出一个命题。

命题 12 设 $(p\mathcal{B}, f, \sigma)$ 和 $(p\mathcal{B}', f', \sigma')$ 是两个 dra。我们可以在双指数时间内判定 $(p\mathcal{B}, f, \sigma)$ 和 $(p\mathcal{B}', f', \sigma')$ 是否等价。

证明。等式 $p\mathcal{B} = p\mathcal{B}'$ 可以在线性时间内判定，而等价关系 $f \equiv f'$ 可以在双指数时间内判定（参见 [29] 中的定理）。最后，根据推论 11，可以在双指数时间内判定实现 $\sigma$ 和 $\sigma'$ 的可信性，因此我们得出结论。 ■

最后，我们引入实现部分可信性的概念。具体而言，设 $(p\mathcal{B}, f, \sigma)$ 是一个 dra。那么，如果对于每个 $l \geq 0$，都存在 $\gamma'_l \subseteq \gamma(\sigma_l)$ 和一个自然数 $g \geq 0$，使得对于每个$l \geq g$都有 $\gamma'_l \models f$，则称实现 $\sigma$ 对于 $f$是部分可信的。显然，对于 $f$ 可信的每个实现，也都对于 $f$部分可信。不过，正如我们将在下文所示，我们可以判定一个不可信的实现是否部分可信。为此，我们需要一些预备知识。

设 $p\mathcal{B}$ 是一个基于组件的系统，$f = \bigsqcup_{i\in I} \sum_{i\in J_i} m_{i,\,j}$ 是定义在 $P_{p\mathcal{B}}$ 上的一个完全范式 PCL 公式。显然，每个完全单项式 $m_{i,\,j}$ 都唯一确定一个交互 $a_{i,\,j}$，对于每个 $i \in I$ 和 $j \in J_i$ 均如此。接下来，设 $\gamma \in C_{p\mathcal{B}}$ 满足 $\gamma \not\models f$。因此，对于每个 $i \in I$，都有 $\gamma \not\models \sum_{j\in J_i} m_{i,\,j}$，进而可推出，对于每个 $i \in I$，都有 $\gamma \neq \{a_{i,j} \in I_{p\mathcal{B}} \mid j \in J_i\}$。我们令 $\mathcal{D}_f = \{\gamma \in C_{p\mathcal{B}} \mid \{a_{i,j} \mid j \in J_i\} \subseteq \gamma$，其中 $i \in I\}$。

命题 13 设 $(p\mathcal{B}, f, \sigma)$ 是一个 dra，且对于 $f$， $\sigma$ 不可信。那么，我们可以判定 $\sigma$ 对于 $f$ 是否部分可信。

证明。考虑到定理 10 的证明，我们计算集合 $\mathcal{D}_{f'}$。由于我们已经计算出 $C_{p\mathcal{B}}$ 和 $C_{p\mathcal{B}}(\models f')$，这只需要常数时间。我们构造 Büchi 自动机 $\mathcal{B}_{f'} = (\{q'_0, q'_1\}, C_{p\mathcal{B}}, \{q'_0\}, \Delta, \{q'_1\})$，其中

$$\Delta = \{(q'_0, \gamma, q'_0) \mid \gamma \in C_{p\mathcal{B}}\} \cup \{(q'_0, \gamma, q'_1) \mid \gamma \in \mathcal{D}_{f'}\} \cup \{(q'_1, \gamma, q'_1) \mid \gamma \in \mathcal{D}_{f'}\}.$$

我们通过与定理 10 和推论 11 中类似的论证得出结果。 ■

需要指出的是，判定 $\sigma$ 的部分可信性不需要额外时间，因为根据定理 10，我们已经证明 $\sigma$ 在双指数时间内是不可信的。此外，最后一个结果使我们能够“修正”一个部分可信的实现。具体来说，沿用上述记号，我们考虑实现 $\sigma' = \left(\sigma_l|_{\gamma'_l}\right)_{l\geq 0}$，它对每个 $l \geq 0$和 $a \in I_{p\mathcal{B}}$都定义为：若 $a \in \gamma'_l$，则 $\sigma_l|_{\gamma'_l}(a)$ =激活；否则 $\sigma_l|_{\gamma'_l}(a)$ = 未激活。显然，实现 $\sigma'$ 现在对 $f$是可信的。

未来工作。
i) 我们的“修正”方法的复杂度有待计算。
ii) 实现 $\sigma$ 可能不可信，原因如下：(a) 它部分可信（见上文）；(b) 存在无穷多个 l，使得 $\gamma$ ($\sigma_l$)中缺少所需交互；或 (c) (a) 和 (b) 两种情况无限次组合。要确定实现的不可信是由情况 (b) 还是 (c) 导致的，甚至找到一种“修正”它的方法，都是一项挑战。

iii) 开发一种实现我们理论的软件工具应该会很有意义 y.
iv) 我们认为，研究由 PCL 描述的可重构架构的模型检验技术十分重要。在 [19] PCLis不断演化，但其公式作为原子命题参与线性时序逻辑（LTL）。

## 参考文献

[1] V.S. Alagar、K. Periyasamy，《B 方法》，载于《软件系统规范》，计算机科学丛书，施普林格，伦敦，2011。doi:10.1007/978-0-85729-277-3 19[2] R. Allen、R. Dɵuence、D. Garlan，动态软件架构的规范与分析，载于：E. Astesiano（编），FASE 1998 论文集，LNCS 1382（1998），21–37。doi:10.1007/BFb0053581[3] S. Amaro、E. Pimentel、A.M. Roldan，基于 REO 的交互模型，《电子版理论计算机科学札记》160（2006），3–14。doi:10.1016/j.entcs.2006.05.012[4] R. Baldoni、S. Bonomi、M. Platania、L. Querzoni，基于主题的发布/订阅系统中的动态消息排序，载于：第 26 届国际并行与分布式处理研讨会论文集，IEEE（2012），第 909–920 页。doi:10.1109/IPDPS.2012.86

[5] R.E. Ballouli、S. Bensalem、M. Bozga、J. Sifakis，动态可重构系统编程，《软件工具与技术转移国际期刊》23（2021），701–719。doi: 10.1007/s10009-020-00596-7[6] L. Baresi、C. Ghezzi、L. Mottola，关于发布-订阅架构的精确验证，载于：ICSE 2007 论文集，IEEE，第 199–208 页。doi: 10.1109/ICSE.2007.57[7] S. Bliudze、J. Sifakis，连接器代数——构建 BIP 中的交互。《IEEE计算机汇刊》57(10)（2008），1315–1330。doi:10.1109/TC.2008.2[8] M. Bozga、R. Iosif、J. Sifakis，关于参数化和可重构的基于组件的系统的局部推理。2019。hal-02267423[9] R. Bruni、A. Bucchairone、S. Gnesi、H. Melgratti，使用类型化图文法对动态软件架构建模，《电子版理论计算机科学札记》213（2008），39–53。doi:10.1016/j.entcs.2008.04.073[10] A. Bucchiarone、J.P. Galeotti，使用 DynAlloy 验证动态软件架构，载于：C. Ermel、R. Heikeil、J. de Lara（编），第七届图转换与可视化建模技术国际会议（GT-VMT 2008）论文集，《EASST电子通讯》10（200）。[11] F. Buschmann、R. Meunier、H. Rohnert、P. Sommerlad、M. Stal，《面向模式的软件架构：模式系统》。Wiley软件模式丛书，1996。[12] E. Cavalcante、T. Batista、F. Oquendo，支持动态软件架构：从架构描述到实现，载于：L. Bass、P. Lago、P Kruchten（编），WISCA 2015 论文集，IEEE（2015），第 31–40 页。doi:10.1109/WICSA.2015.21[13] D.D. Corkill，黑板系统，《AI Expert》6(9)（1991），40–47。[14]P. Eugster、P. Felber、R. Guerraoui、A.-M. Kermarrec，发布/订阅的多种面貌，《ACM计算调查》35(2)（2003），114–131。doi:10.1145/857076.857078[15] J.L Fiadeiro、A. Lopes，面向服务架构中的动态重配置模型，《软件与系统建模》12（2013），349–367。doi:10.1007/s10270-012-0236-1[16] D. Garlan，软件架构：一段游记，载于：FOSE 2014 论文集，ACM（2014），29–39。doi:10.1145/2593882.2593886[17] D. Garlan、M. Shaw，软件架构导论，载于：《软件工程与知识工程进展》，世界科学出版社，1993，第 1–39 页。doi:10.1142/9789812798039 0001[18]N. He 等，X-MAN 中基于组件的设计与验证，载于：ERTS$^2$ *2012*。获取地址：https://web1.see.asso.fr/erts2012/Site/0P2RUC89/1D-2.pdf

-

[19] A. El-Hokayem, M. Bozga, J. Sifakis，面向动态可重构系统的时序配置逻辑，载于：C.-C. Hung、J. Hong、A. Bechini、E. Song（编），《SAC '21论文集：2021年第36届ACM/SIGAPP应用计算研讨会》，ACM，2021年，第1419–1428页。doi:10.1145/3412841.3442017[20]T. Han、T. Chen、J. Lu，基于空间逻辑的动态软件架构结构分析，载于：《COMPSAC 2005论文集》，IEEE，2005年，第71–76页。doi:10.1109/COMPSAC.2005.146[21] V. Karyoti、P. Paraponiari，乘积赋值幺半群上的加权PCL，载于：S. Bliudze、L. Bocchi（编），《COORDINATION 2020论文集》，LNCS 12134（2020），第301–319页。doi:10.1007/978-3-030-50029-0 19[22] J. Magee、J. Kramer，动态软件架构，载于：SIGSOFT '96，ACM，1996年，第3–14页。[23] M.S. Mansureh、J.-M. Cho、K. Choi，可重构架构，载于：S. Ha、J. Teich（编），《硬件/软件协同设计手册》，第1–42页。doi:10.1007/978-94-017-7358- 4 12-1[24]A. Mavridou、E. Baranov、S. Bliudze、J. Sifakis，配置逻辑：架构风格建模，《J. Log. Algebr. Methods Program.》86（2016），第2–29页。doi:10.1016/j.jlamp.2016.05.002。[25] N. Medvidovic，架构描述语言与动态架构变更，载于：A.L. Wolf、A. Finkelstein、G. Spanoudakis、L. Vidal（编），《第二届国际软件架构研讨会（ISAW-2）与国际软件开发多视角研讨会（Viewpoints '96）联合论文集：SIGSOFT '96研讨会》，ISAW '96，美国加利福尼亚州旧金山，ACM，1996年，第24–27页。doi:10.1145/243327.243340[26] C. Metzner、L. Cortez、D. Chacin，在Web应用中使用黑板架构，载于：《 $I^nSITE\ 2005$》，《信息科学与信息技术问题》3（2005）。doi:10.28945/2929[27] H.P. Nii，黑板系统，第一部分：问题求解的黑板模型及黑板架构的演进，《AI Mag.》7（2）（1986），第38–53页。[28] A. Olivieri、G. Rizzo、F. Morand，物联网集成的发布-订阅方法：智能办公室用例，载于：L. Baroli、M. Takizawa、F. Xhafa、T. Enokido、J.H. Park（编），《第29届国际高级信息网络与应用研讨会论文集》，IEEE，2015年，第644–651页。doi:10.1109/WAINA.2015.28[29] P. Paraponiari、G. Rahonis，加权命题配置逻辑：具有定量特征的架构规约语言，《Inform. Comput.》282（2022），104647。doi:10.1016/j.ic.2020.104647[30] S. Patel、S. Jardosh、A. Makwana、A. Thakkar，物联网发布/订阅机制：事件匹配算法综述与开放研究挑战，载于：N. Modi、P. Verma、B. Trivedi（编），《国际通信与网络会议论文集》，《智能系统与计算进展》508（2017），第287–294页。doi:10.1007/978-981-10-2750-5 30

[31] M. Pittou、G. Rahonis，参数化组件式系统中的架构：定性与定量建模，《Log. Methods Comput. Sci.》17（4）（2021），16:1–26:53。doi:10.46298/lmcs-17(4:26)2021[32] B. Seli´c，动态软件架构规约，《软件与系统建模》20（2021），第595–605页。doi:10.1007/s10270-021-00875-0[33] J. Straub、H. Reza，黑板架构在用于控制具有不同执行器和运动能力的飞行器的决策系统中的应用，载于：S. Latifi（编），《第11届新一代信息技术国际会议论文集》，ITNG 2014，IEEE计算机学会，第514–519页。[34] K. Yang、K. Zhang、X. Jia、M.A. Hasan、X Shen，云平台上隐私保护的基于属性关键词的数据发布-订阅服务，《Inform. Sci.》387（2017），第116–131页。doi:10.1016/j.ins.2016.09.020[35]K. Zhang、V. Muthusamy、H.-A. Jacobsen，基于内容的发布/订阅系统中的全序，载于：《第32届分布式计算系统国际会议论文集》，IEEE（2012），第335–344页。doi:10.1109/ICDCS.2012.17