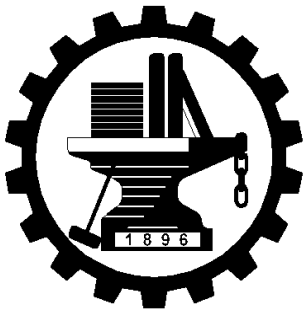


博士论文答辩

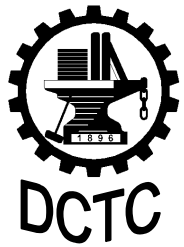
鲁棒灰箱演算的类型系统和代数性质研究

博士生：管旭东
导师：尤晋元

上海交大计算机科学与工程系
分布计算技术中心

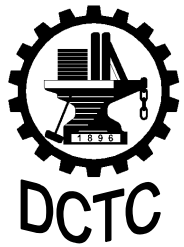
2002 . 6

内容提要



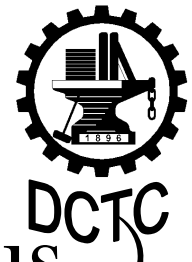
1. 论文的背景
2. 灰箱演算，研究ROAM的原因
3. 演化类型系统
4. ROAM的进程等价性
5. π演算的翻译及证明
6. 总结论文的创新处

论文的背景



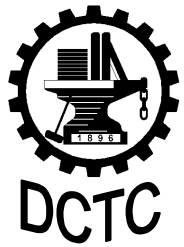
- 移动计算
  - ⇨ 硬件移动、软件移动
  - ⇨ 网络的广域化、全球化
  - ⇨ 从技术到理论支持
- 从程序设计理论（ programming language theory ）角度
  - ⇨ 寻求支持并发**(concurrency)**、分布**(distribution)**、移动**(mobility)**概念的核心模型（ **CCS,CSP,π,χ,join,ambient,seal,...** ）
  - ⇨ 目标：作为支持全球性移动计算编程语言的理论基础

相关研究领域



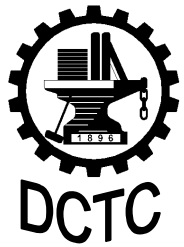
- 本论文有关灰箱演算（ ambient calculus ）的研究工作，属于：
  - ⇨ 程序设计理论（ 并发、分布、移动 ）
  - ⇨ 进程代数
- 相关研究领域：
  - ⇨ **OS (distributed/mobile): OS support for Java and mobile agent technology**
  - ⇨ **TCS: type theory and semantics**
  - ⇨ **Security - Wide area network, software mobility**

灰箱演算的相关工作



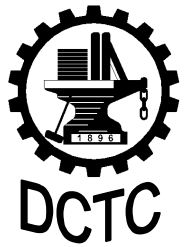
- 支持分布、移动的形式化模型，除了Ambient, 还有：
  - ⇨ **λdist: Sekiguchi,et.al., 1997, U.Tokyo**
  - ⇨ **Dπ: Henessey, et.al., 1998, U. Sussex**
  - ⇨ **π_{1l} : Amadio, 1997**
  - ⇨ **Distributed-π: Sewell, 1998, Cambridge**
  - ⇨ **D-join: Fournet, et.al., 1996, INRIA**
  - ⇨ **Seal: Vitek, et.al, 1999, U. Purdue**
  - ⇨ **...**

内容提要



1. 论文的背景
2. 灰箱演算，研究ROAM的原因
3. 演化类型系统
4. ROAM的进程等价性
5. pi演算的翻译及证明
6. 总结论文的创新处

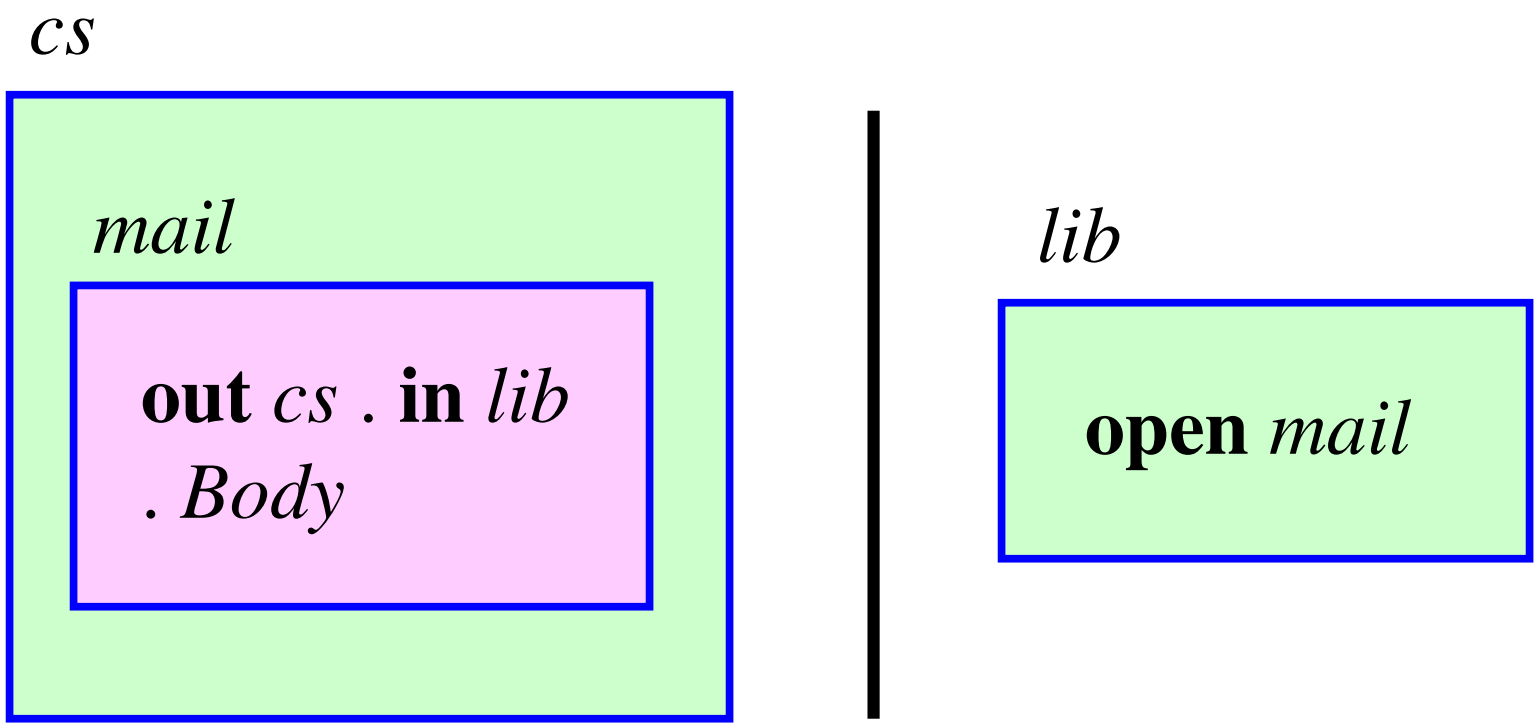
MA在提出后迅速成为研究热点



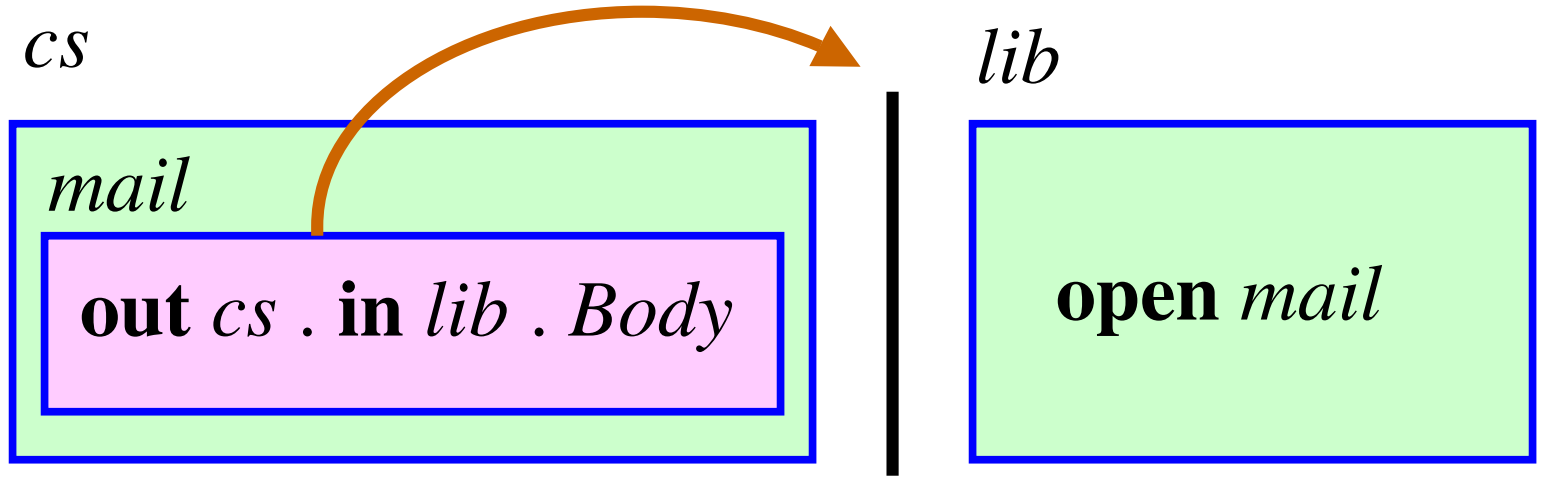
- Cardelli & Gordon: “Mobile Ambients”
  - ⇨ **Digital Research => Microsoft Research**
  - ⇨ 会议：**FoSSaCS’98**
  - ⇨ 期刊：**Theoretical Computer Science, 2000**
- 后续研究：
  - ⇨ **Ambient有关的研究论文：50多篇**
  - ⇨ **FoSSaCS’98的引用数：200左右 (http://citeseer.nj.nec.com/cardelli98mobile.html， 2002-05) (SCI引用数: 50左右)**
  - ⇨ 欧盟：**2002启动FET-Global Comp.计划**

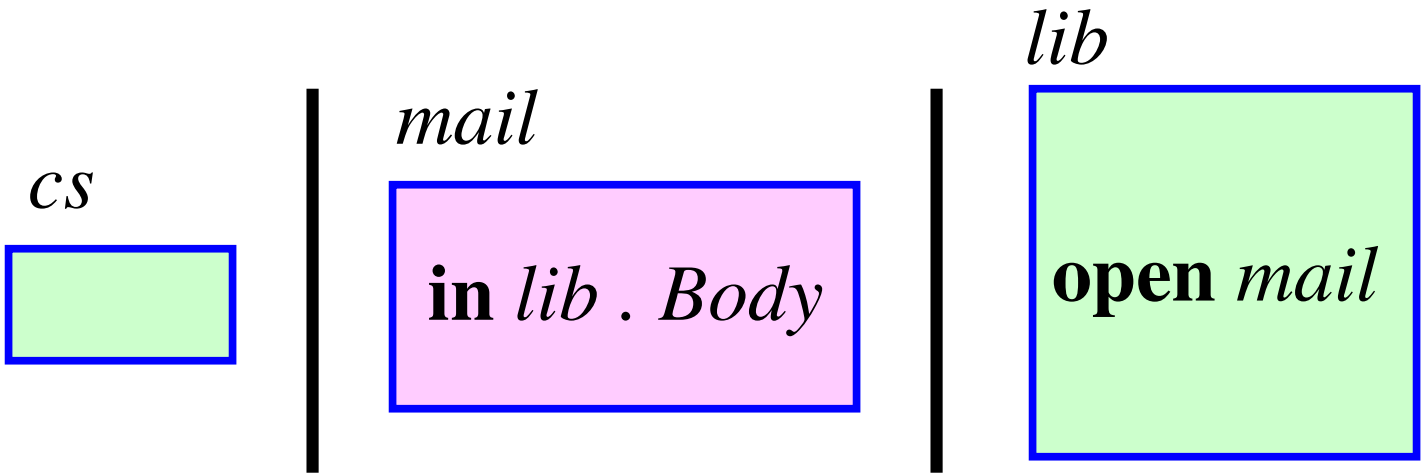
MA用三个原语来描述移动计算

cs[*mail*[**out** *cs* . **in** *lib* . *Body*]] | *lib*[**open** *mail*]

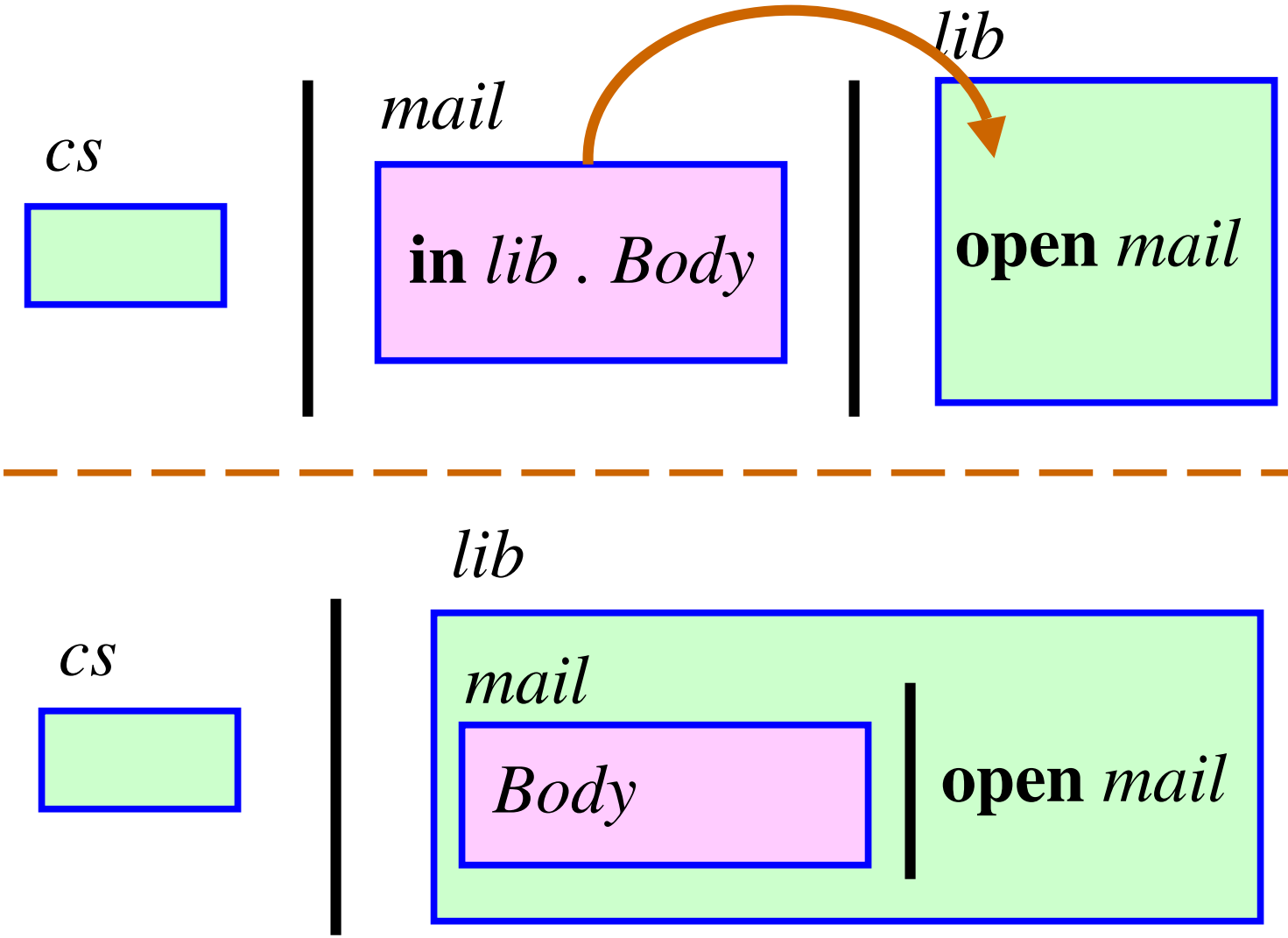


out 使一个灰箱移出其父灰箱

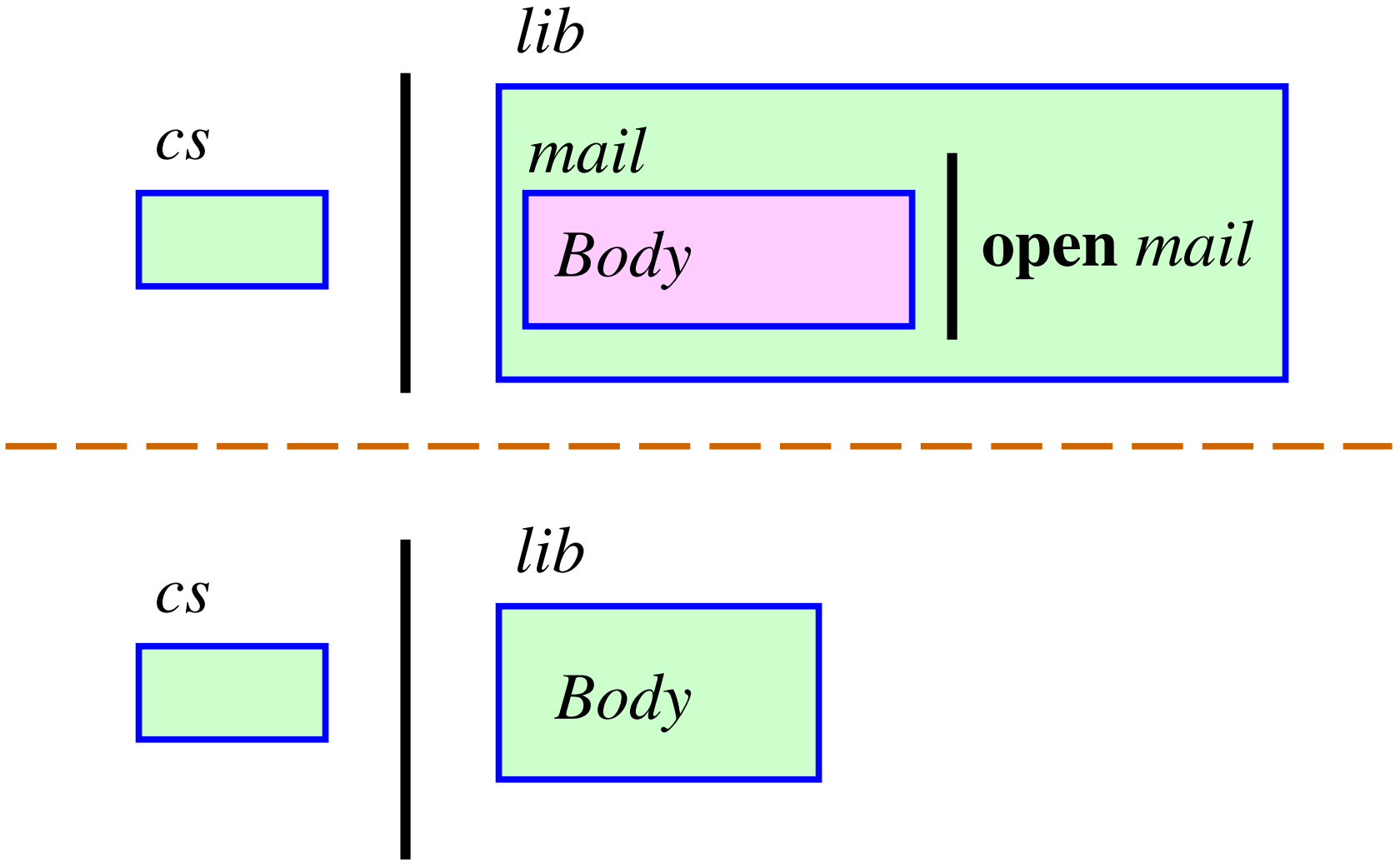




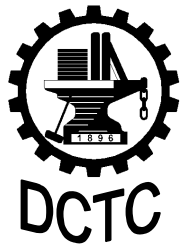
in 使一个灰箱移进另一个同层灰箱



open 用来打开一个灰箱



防火墙跨越体现了MA的安全性 (page72)



防火墙的一般形式： $(\nu w)(w[\ \ldots\ ])$

授信代理(*Agent*)跨越防火墙(*Firewall*)：

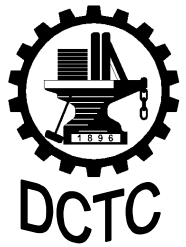
$$Agent = m[\mathbf{open}\ k\ .\ k'\ [Q]]$$

$$Firewall = (\nu w)(w[k[\mathbf{out}\ w\ .\ \mathbf{in}\ m\ .\ \mathbf{in}\ w] \mid \mathbf{open}\ m\ .\ \mathbf{open}\ k'\ .\ P])$$

$$Agent \mid Firewall \rightarrow^{*} (\nu w)(w[P \mid Q]\ )$$

$$(\nu m\ k\ k')(Agent \mid Firewall)\ \approx\ (\nu w\ m\ k\ k')(w[P \mid Q]\ )$$

防火墙跨越前后的等价性推导(GC99)



$$(\nu\, m\ k\ k')\ (Agent \mid Firewall)$$

$$= (\nu\, m\ k\ k')\ (m[\mathbf{open}\ k\ .\ k'\ [Q]] \mid (\nu\, w)(w[k[\mathbf{out}\ w\ .\ \mathbf{in}\ m\ .\ \mathbf{in}\ w] \mid \mathbf{open}\ m\ .\ \mathbf{open}\ k'\ .\ P]))$$

$$\approx (\nu\, m\ k\ k'\ w)\ (m[\mathbf{open}\ k\ .\ k'\ [Q]] \mid k[\mathbf{in}\ m\ .\ \mathbf{in}\ w] \mid w[\mathbf{open}\ m\ .\ \mathbf{open}\ k'\ .\ P])$$

$$\approx (\nu\, m\ k\ k'\ w)\ (m[\mathbf{open}\ k\ .\ k'\ [Q] \mid k[\mathbf{in}\ w]] \mid w[\mathbf{open}\ m\ .\ \mathbf{open}\ k'\ .\ P])$$

$$\approx (\nu\, m\ k\ k'\ w)\ (m[k'\ [Q] \mid \mathbf{in}\ w] \mid w[\mathbf{open}\ m\ .\ \mathbf{open}\ k'\ .\ P])$$

$$\approx (\nu\, m\ k\ k'\ w)\ (w[m[k'\ [Q]] \mid \mathbf{open}\ m\ .\ \mathbf{open}\ k'\ .\ P])$$

$$\approx (\nu\, m\ k\ k'\ w)\ (w[k'\ [Q] \mid \mathbf{open}\ k'\ .\ P])$$

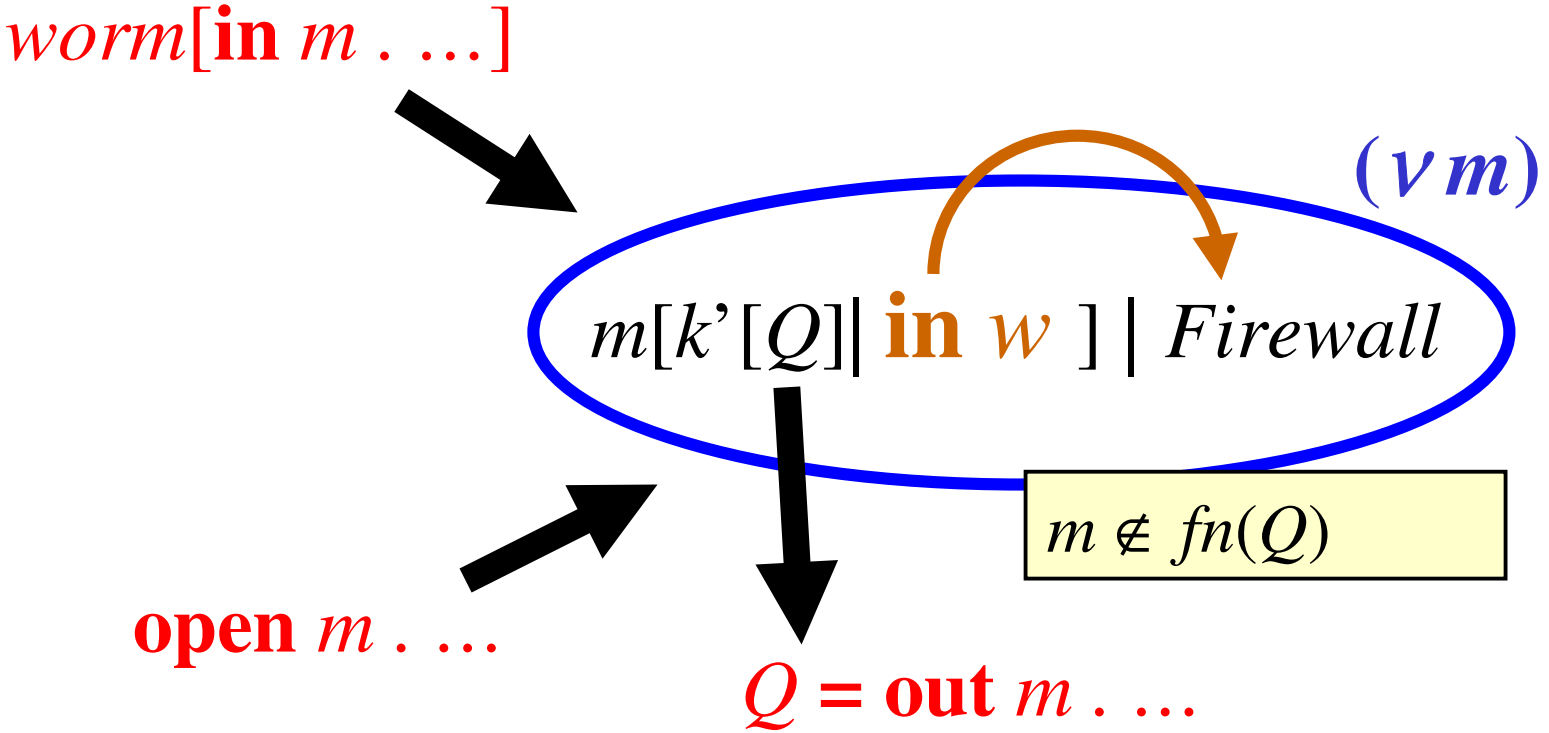
$$\approx (\nu\, m\ k\ k'\ w)\ (w[Q \mid P])$$

通过约束实现并发、移动进程行为的确定性

MA存在强干扰问题

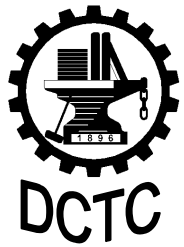
一旦失去约束，进程行为将不可预测！

以取消对*m*的约束为例：



后果：*m*的约束将限制其在防火墙外的功能

SA演算通过增加协动作解决强干扰



Mobile Safe Ambient (SA):

增加协动作，控制强干扰

in – $\overline{\text{in}}$, out – $\overline{\text{out}}$, open – $\overline{\text{open}}$

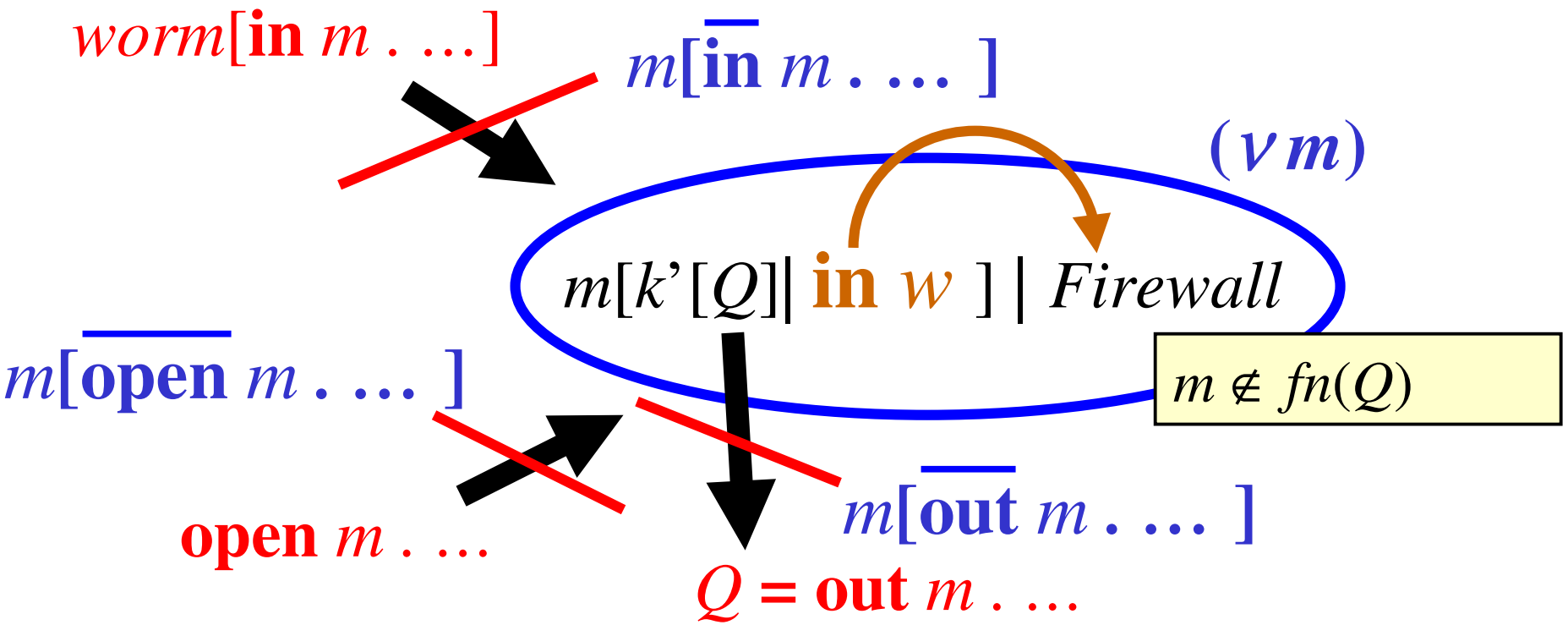
SA中的可归约形式：

移入： $m[\mathbf{in}\ w\ .\ \ldots] \mid w\ [\overline{\mathbf{in}}\ w\ .\ \ldots]$

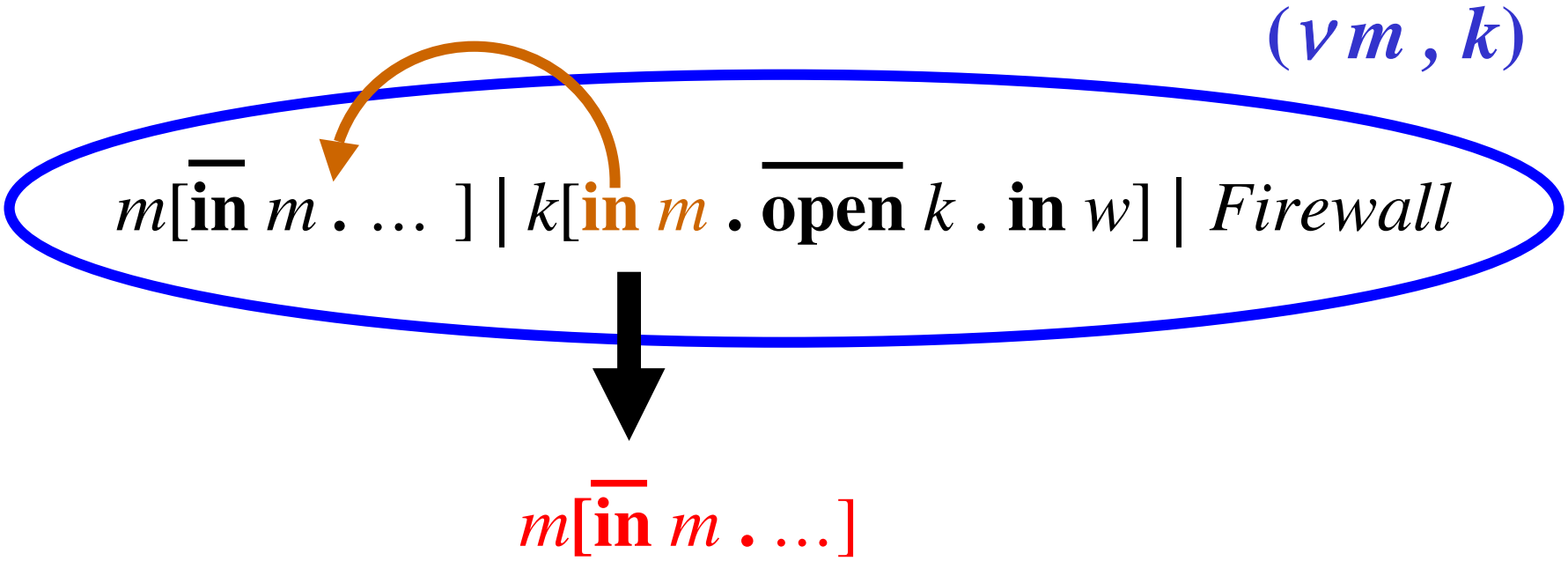
移出： $m[k[\mathbf{out}\ m\ .\ \ldots] \mid \overline{\mathbf{out}}\ m\ .\ \ldots]$

打开： $\mathbf{open}\ m \mid m\ [\overline{\mathbf{open}}\ m\ .\ \ldots]$

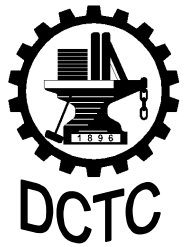
SA减少了防火墙跨越等价性的条件



但是对*m*的约束依然存在



ROAM修改了SA协动作参数的语义



Robust Ambient (ROAM):

在协动作基础上，进一步利用其参数提高安全性，减少二义性

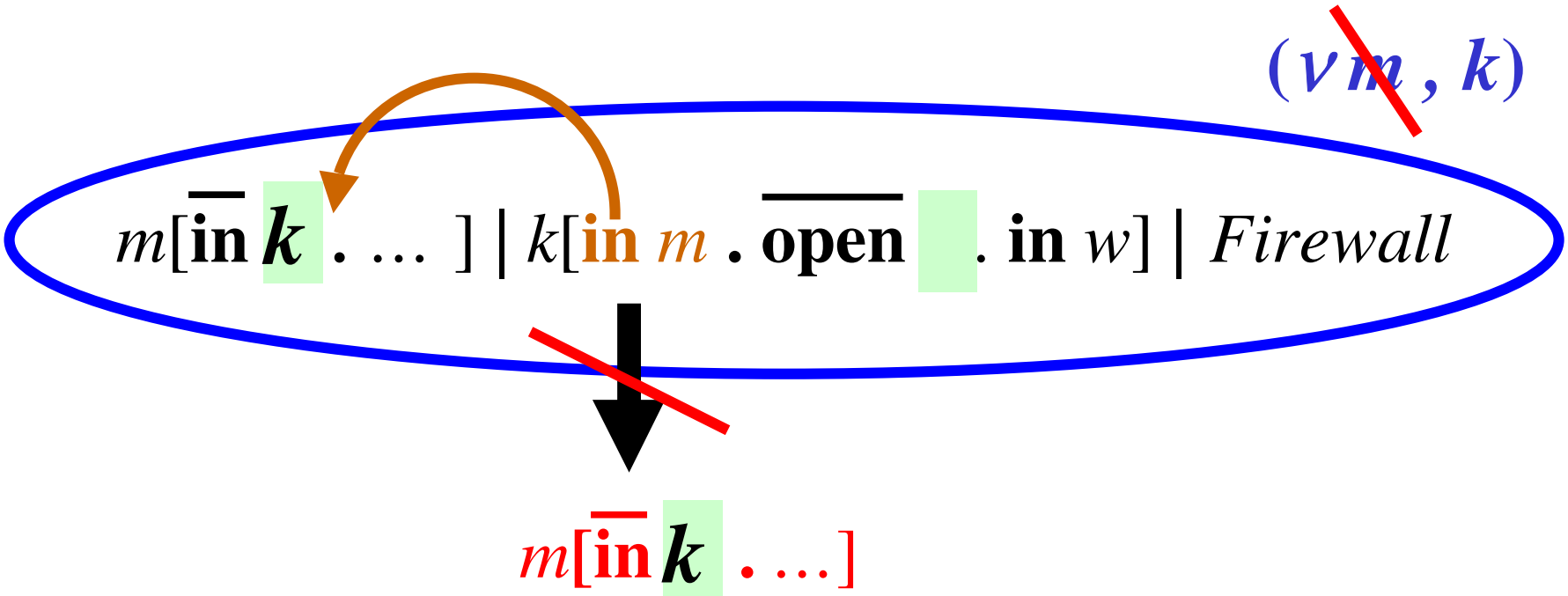
ROAM 中的可归约形式：

移入： $m[\mathbf{in}\ w\ .\ \ldots] \mid w\ [\overline{\mathbf{in}}\ \boldsymbol{m}\ \ldots]$

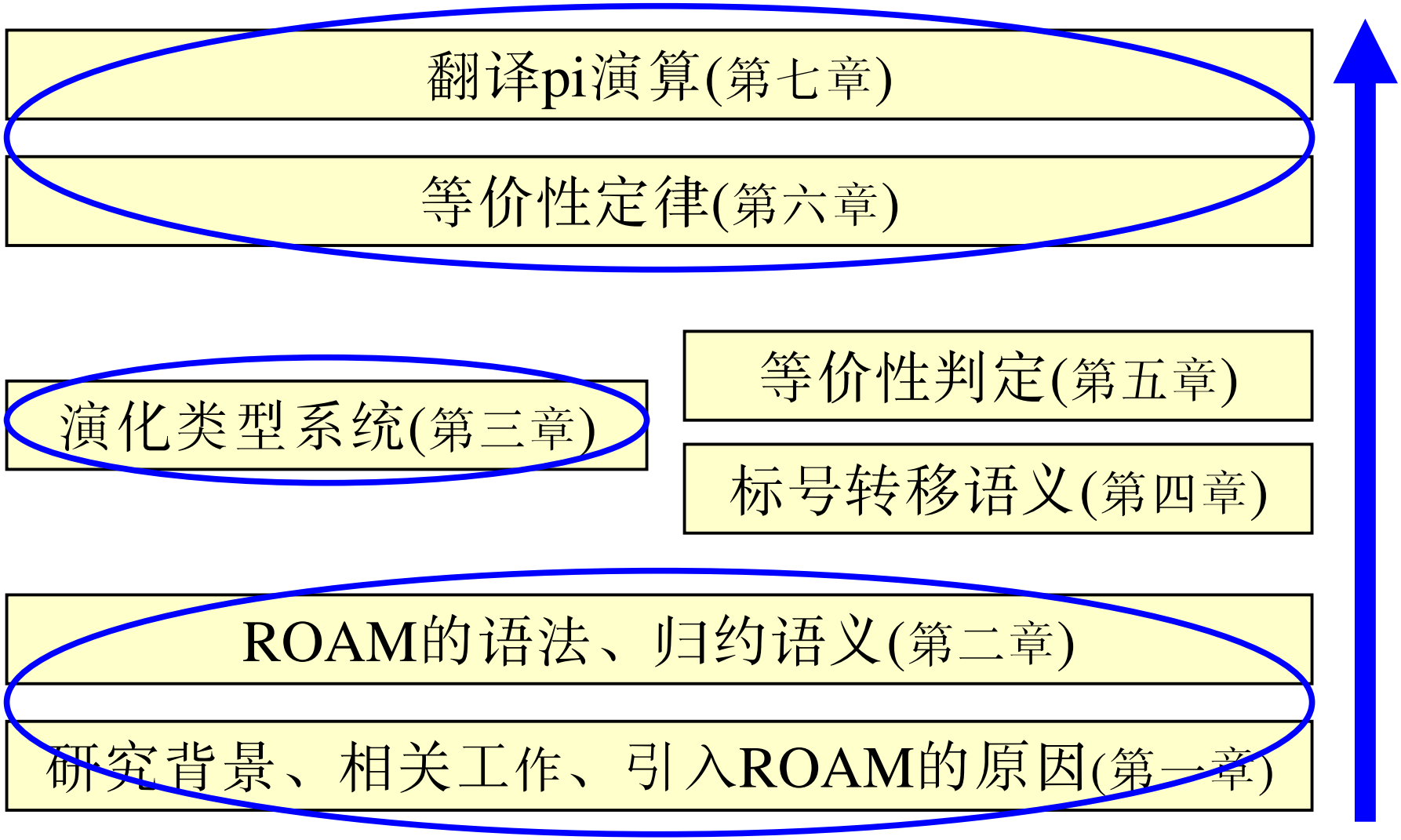
移出： $m[k[\mathbf{out}\ m\ .\ \ldots] \mid \overline{\mathbf{out}}\ \boldsymbol{k}\ .\ \ldots]$

打开： $\mathbf{open}\ m \mid m\ [\overline{\mathbf{open}}\ \ .\ \ldots]$

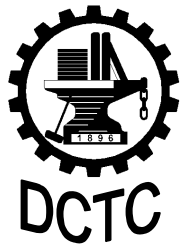
ROAM完全取消了对*m*的约束



论文内容的依赖关系

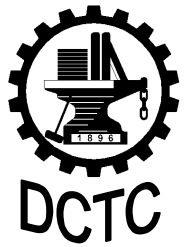

翻译pi演算(第七章)
等价性定律(第六章)
演化类型系统(第三章)
等价性判定(第五章)
标号转移语义(第四章)
ROAM的语法、归约语义(第二章)
研究背景、相关工作、引入ROAM的原因(第一章)

内容提要



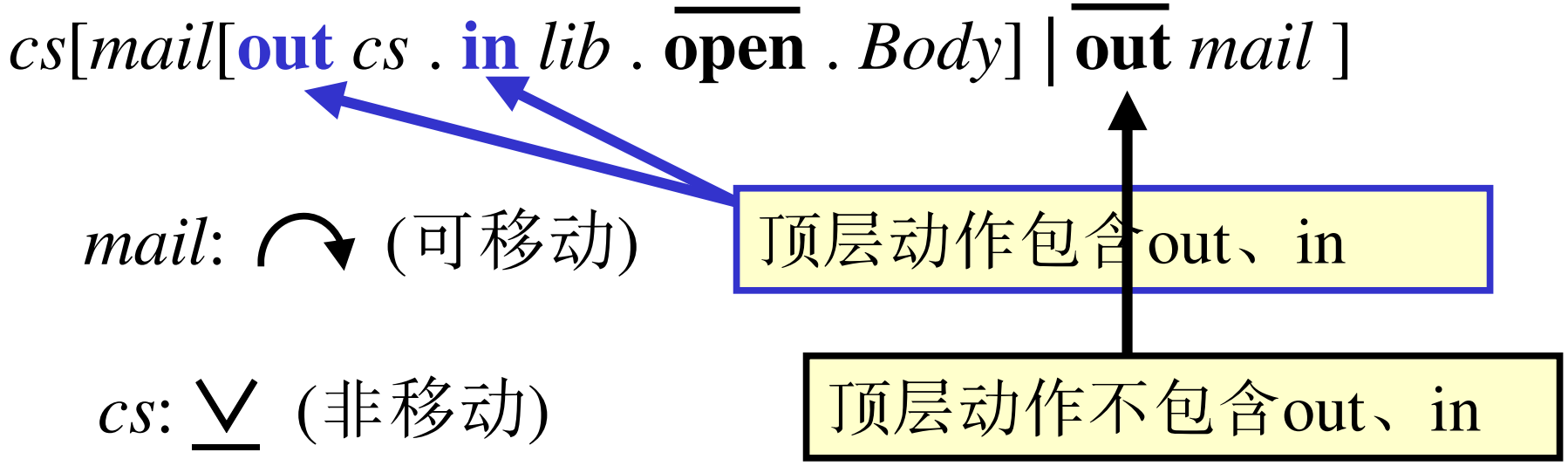
1. 论文的背景
2. 灰箱演算，研究ROAM的原因
3. 演化类型系统
4. ROAM的进程等价性
5. π演算的翻译及证明
6. 总结论文的创新处

类型系统可以限定进程的属性

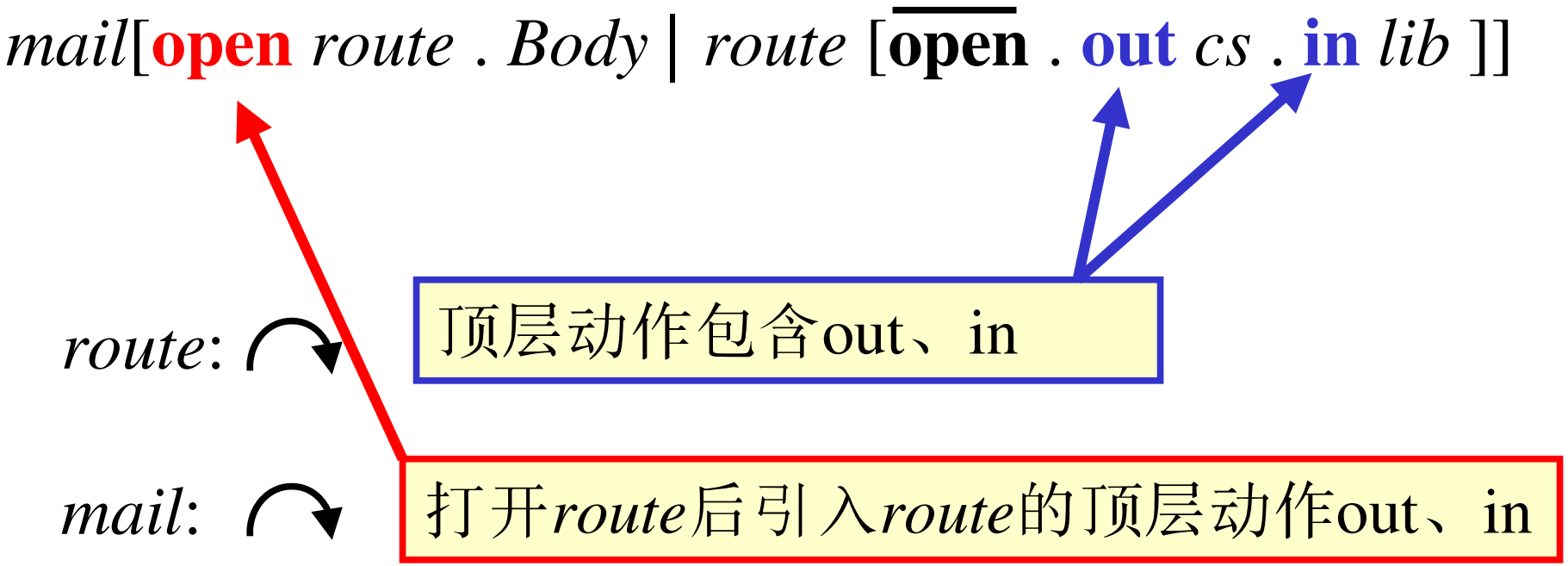


- 传统程序语言：
  - ⇨ 数据类型：变量是整数还是指针？
  - ⇨ 函数类型：函数的参数和返回结果类型？
- 灰箱演算：
  - ⇨ 移动性：灰箱会移动吗？
  - ⇨ 安全性：灰箱的活动范围是什么？会进入受保护的禁区吗？会被谁打开？
  - ⇨ 消息传递类型：灰箱中允许广播何种类型的数据？
  - ⇨ 线程数：进程可以激发多少个并发动作？
  - ⇨ …

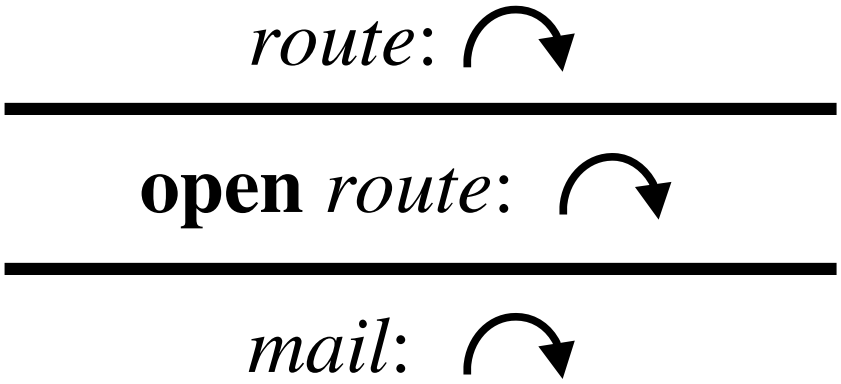
Fact 1: 灰箱移动性取决于它的顶层动作



Fact 2: 打开动作将引入被打开灰箱的顶层动作



综合上述两点，易得以下结论：

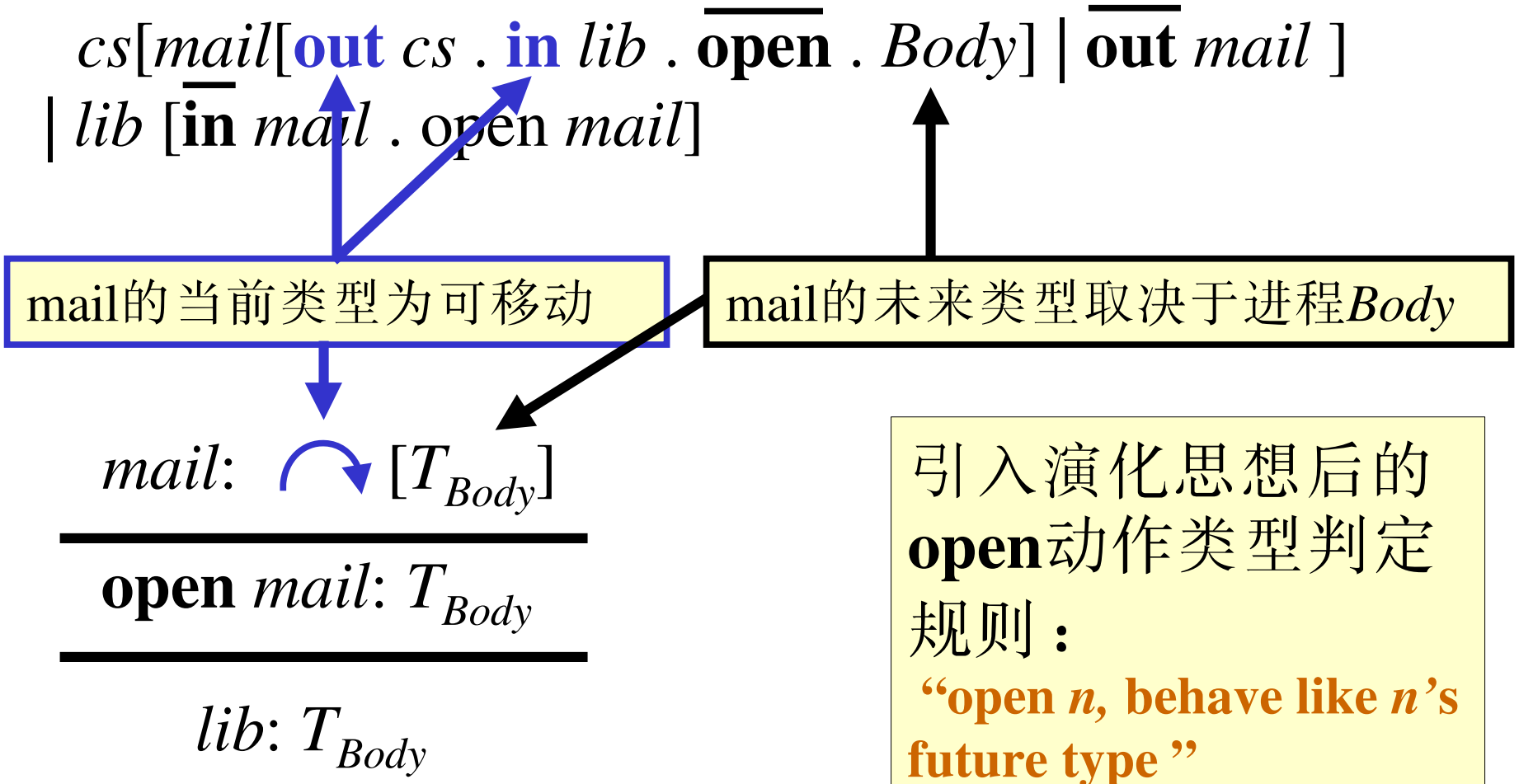


传统的**open**动作类型判定规则：
“open *n*, behave like *n*”

- 使用该规则的文献：
  - ⇨ **Cardelli & Gordon: POPL’99, ICALP’99**
  - ⇨ **Zimmer: FoSSaCS’2000**
  - ⇨ **…**

对open动作类型判定规则的细化

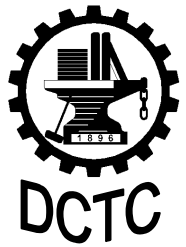
Revision of Fact 2: 打开动作将只引入被打开灰箱位于$\overline{\textbf{open}}$动作后的顶层动作



引入演化思想后的**open**动作类型判定规则：

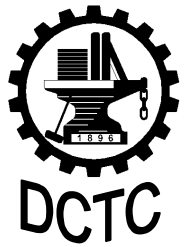
"open *n*, behave like *n*'s future type"

论文第三章详细介绍了演化类型系统



- 主要工作
  - ⇨ 描述了进程的移动性和线程数两个性质
  - ⇨ 区分当前类型和未来类型，支持类型演化
  - ⇨ 支持子类型关系
  - ⇨ 给出最小类型推导算法
- 演化类型系统的作用
  - ⇨ 更细致地刻画进程/灰箱特性
  - ⇨ 为研究进程等价性定律打下基础

内容提要

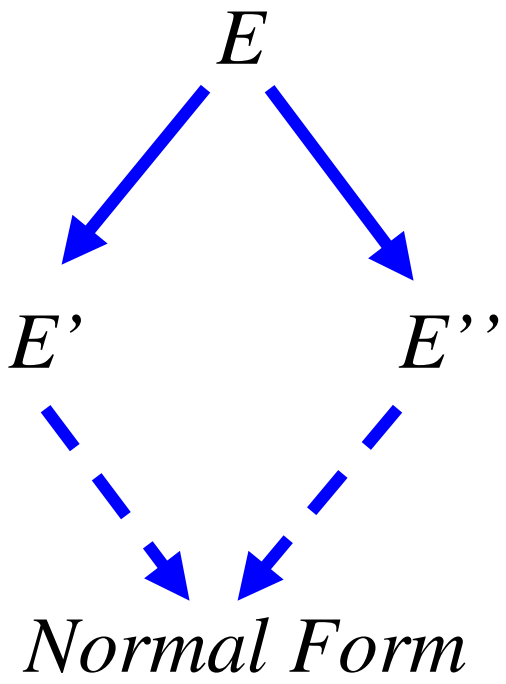


1. 论文的背景
2. 灰箱演算，研究ROAM的原因
3. 演化类型系统
4. ROAM的进程等价性
5. π演算的翻译及证明
6. 总结论文的创新处

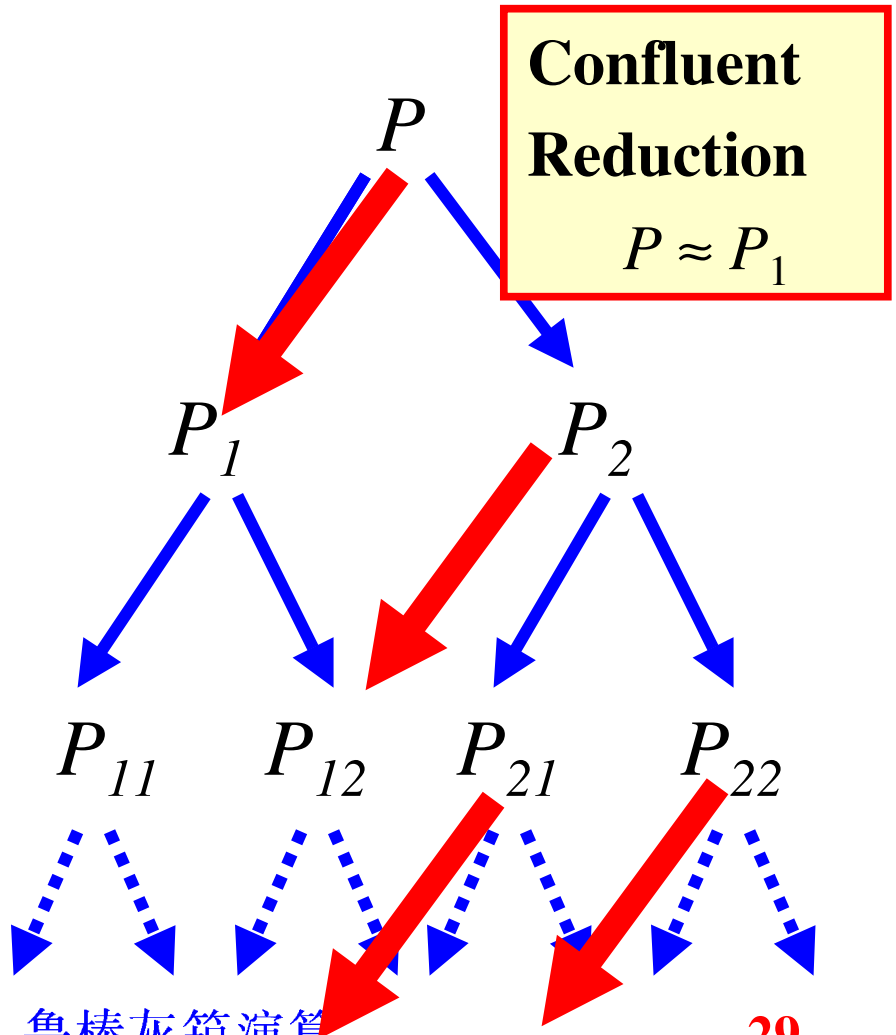
并发进程的等价性研究

顺序执行程序（ 函数表达式归约 ）：

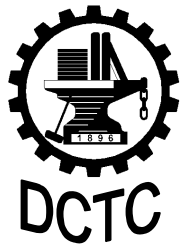
Church-Rosser定理:



并发移动进程：



不同演算的进程等价性条件不同



MA: $(\nu\ n, m)(n[\mathbf{in}\ m\ .\ P] \mid m[\ ]) \mid Q$

$\approx (\nu\ n, m)\ m[\ n[P]\ ] \mid Q$

进程会使m移动

SA: $(\nu\ m)(n[\mathbf{in}\ m\ .\ P] \mid m[\overline{\mathbf{in}}\ m\ .\ P'\ ]) \mid Q$

$\approx (\nu\ m)\ m[\ n[\ P\ ] \mid P'] \mid Q$

无法只约束n

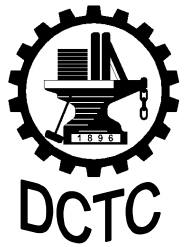
ROAM: $(\nu\ m)(n[\mathbf{in}\ m\ .\ P] \mid m[\overline{\mathbf{in}}\ n\ .\ P'\ ]) \mid Q$

$\approx (\nu\ m)\ m[\ n[\ P\ ] \mid P'] \mid Q$

$(\nu\ n)(n[\mathbf{in}\ m\ .\ P] \mid m[\overline{\mathbf{in}}\ n\ .\ P'\ ]) \mid Q$

$\approx (\nu\ n)\ m[\ n[\ P\ ] \mid P'] \mid Q$

用单线程性可以得到更一般的结论



ROAM: $(\nu\, n)(n[\mathbf{in}\, m\, .\, P] \mid m[\overline{\mathbf{in}}\, n\, .\, P'\,]\,) \mid Q$

$\approx (\nu\, n)\, m[\, n[\, P\,] \mid P'] \mid Q$

如果n, m为单线程，那么：

$(\nu\, n)(n[\mathbf{in}\, m\, .\, P \mid P_1] \mid m[\overline{\mathbf{in}}\, n\, .\, P' \mid P_1'\,]\,) \mid Q$

$\approx (\nu\, n)\, m[\, n[\, P \mid P_1\,] \mid P' \mid P_1'\,] \mid Q$

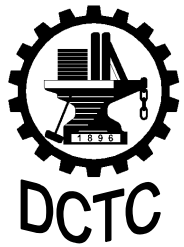
更进一步，如果外部不存在额外的$n[\mathbf{in}\, m\, .\, \ldots]$或$m[\mathbf{in}\, n\, .\, \ldots]$干扰其归约，那么：

$n[\mathbf{in}\, m\, .\, P \mid P_1] \mid m[\overline{\mathbf{in}}\, n\, .\, P' \mid P_1'\,] \mid Q$

$\approx m[\, n[\, P \mid P_1\,] \mid P' \mid P_1'\,] \mid Q$

Page 64, 定理6.17: (ST In)定律

用非移动性可进一步得到定点接收定律



ROAM: $(\nu\ n)(n[\mathbf{in}\ m\ .\ P] \mid m[\overline{\mathbf{in}}\ n\ .\ P'\ ]\ ) \mid Q$

$\approx (\nu\ n)\ m[\ n[\ P\ ] \mid P'] \mid Q$

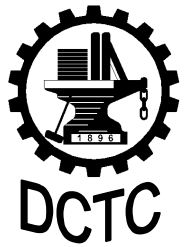
如果n为单线程，m非移动且不能被打开，$\overline{\mathbf{in}}\ n$只在$m$中的复制构造中出现，那么：

$n[\mathbf{in}\ m\ .\ P \mid \boldsymbol{P_1}] \mid m[!\ \overline{\mathbf{in}}\ n\ .\ P' \mid \boldsymbol{P_1'}\ ] \mid Q$

$\approx m[\ n[\ P \mid \boldsymbol{P_1}\ ] \mid !\ \overline{\mathbf{in}}\ n\ .\ P' \mid P' \mid \boldsymbol{P_1'}\ ] \mid Q$

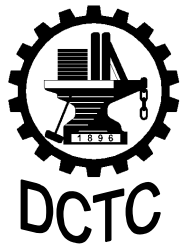
Page 68, 定理6.24: (Rec Co-in)定律

定律的证明从标号转移语义开始



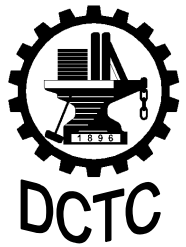
- 第四章：标号转移语义
- 第五章：等价性判定的Context Lemma和Activity Lemma
- 第六章：定价性定律及其证明

内容提要



1. 论文的背景
2. 灰箱演算，研究ROAM的原因
3. 演化类型系统
4. ROAM的进程等价性
5. π演算的翻译及证明
6. 总结论文的创新处

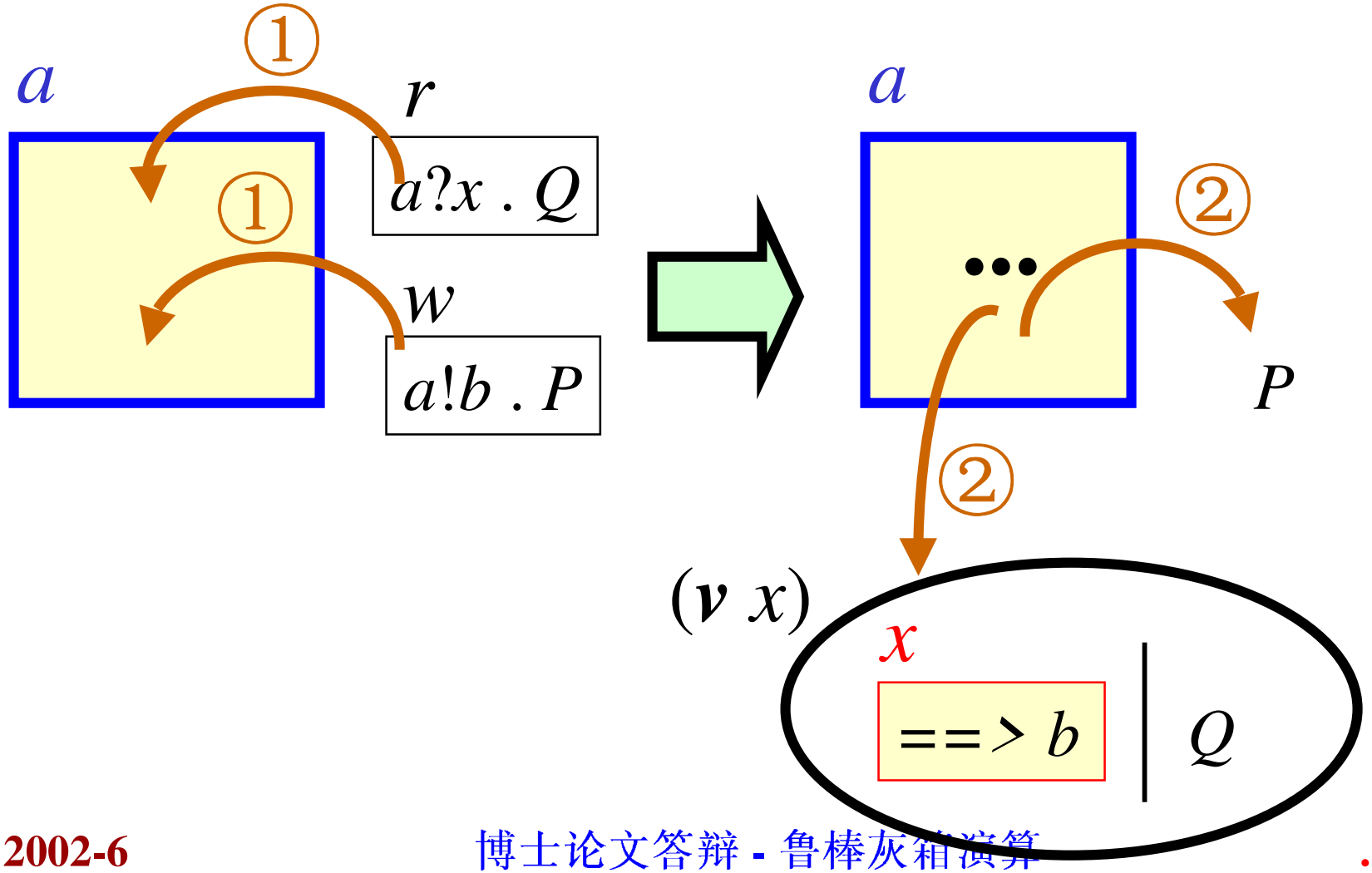
翻译π演算体现了ambient的表达能力



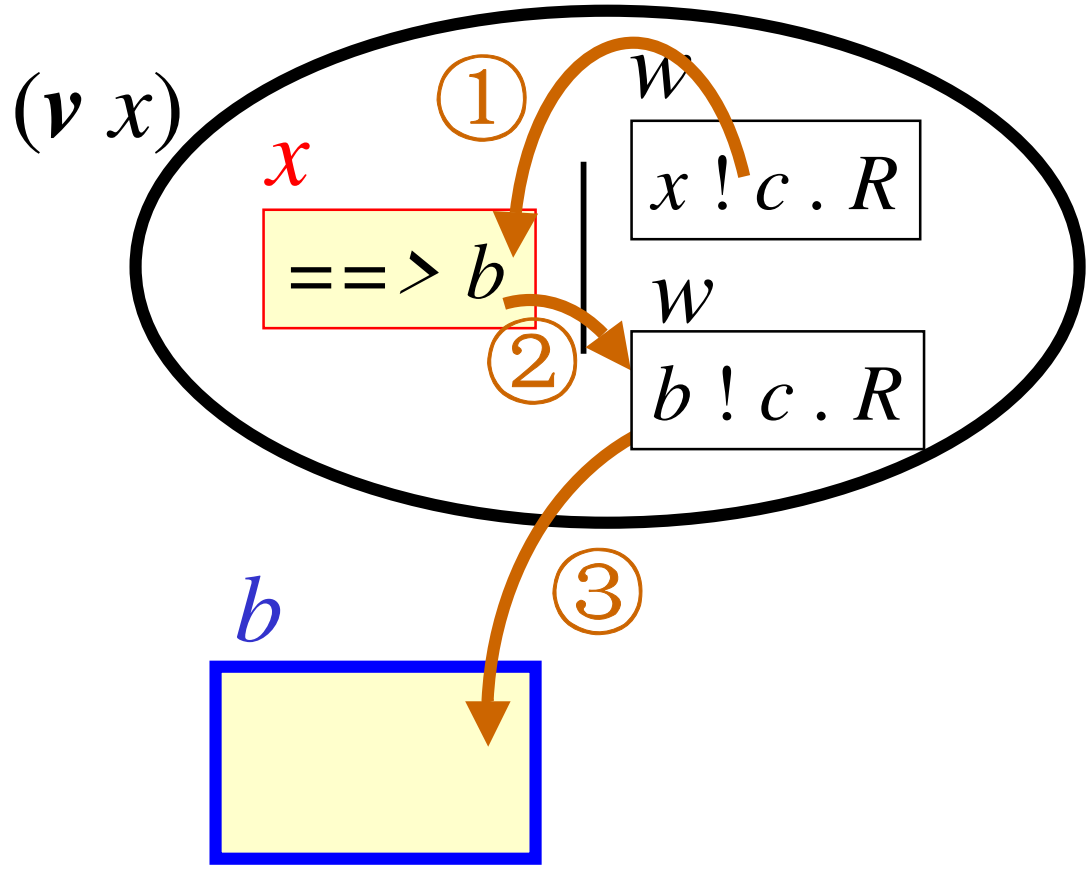
- pi演算的特点
  - ⇨ **channeled communication**
  - ⇨ **name passing and substitution**
- 用带通讯的灰箱演算翻译π
  - ⇨ 直接利用灰箱演算中的本地通讯和替换
- 用不带通讯的灰箱演算翻译π
  - ⇨ **Zimmer, Express'2000**
  - ⇨ 只使用移入、移出和打开三个基本操作
  - ⇨ 建立名字替换服务灰箱，实现**substitution**

输入输出在代表通道的灰箱中进行

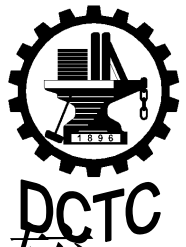
$$a!b\,.\,P \mid a?x\,.\,Q \rightarrow P \mid Q\{b/x\}$$



约束灰箱*x*实现将*Q*中对通道*x*的输入输出重定位到通道*b*

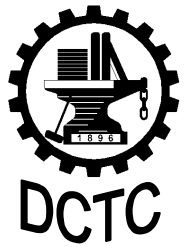


本文对π演算翻译的证明更加简单



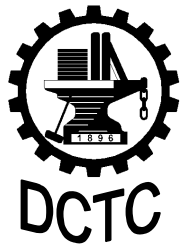
- Zimmer采用直接证明的方法，过程比较复杂
- 本文翻译方案设计时考虑到等价性定律的使用，输入输出灰箱设计为单线程，通道灰箱设计为多线程非移动，证明时可大量使用等价性定律，证明过程大大简化

内容提要



1. 论文的背景
2. 灰箱演算，研究ROAM的原因
3. 演化类型系统
4. ROAM的进程等价性
5. π演算的翻译及证明
6. 总结论文的创新处

论文的创新处



- 利用协动作参数，提出了ROAM
- 提出了灰箱演算的演化类型系统
- 融合ROAM的类型、等价性和表达能力
  - ⇨ 结合演化类型系统，用**Context Lemma**和**Activity Lemma**证明了进程等价性定律
  - ⇨ 用等价性定律证明了对π的翻译