线性逻辑导论

黄林鹏　孙永强　（上海交通大学计算机系）

摘　要

In 1986, J. Y. Girard discovered that usual logical implication could be broken up into more elementary linear operations. Following that, he developed a new logical system, called linear logic, which appears now as a promising approach to fundamental questions arising in proof theory and in computer science. In this paper, we give a brief guide to the characteristics of linear logic, summary the recent developments and discuss the prospect of its applications to computer science.

1. 从古典逻辑到线性逻辑

1986年，法国巴黎第七大学J. Y. Girard教授在研究二阶λ-演算的指称语义时[1]发现古典逻辑中的隐含连接词A⇒B可以分解成更基本的线性运算！A⊸B，在此基础上，一个与证明论及计算机科学密切相关的新型逻辑系统——线性逻辑（LL）诞生了。

由于古典逻辑缺乏切实的方式把证明看成算法，因此是不可构造的，除了不存在任何非平凡的指称语义外，矢列演算中Cut的消去也不满足CR性质。为了避免这些缺点，我们考虑：

1）限制矢列演算的右边公式数（≤1）；

2）限制结构规则的使用；

3）同时施加限制1）和2）。

限制1）导致了直觉主义逻辑的产生；而限制2）事实上就是从中的线性矢列演算；限制3）的结果可称为线性直觉主义逻辑[2]。有趣的是直觉主义逻辑中的公式可以有效地解释到LL。为了更好地理解限制2）的含义，下面我们就公式——资源观点来分析矢列演算中结构规则的作用。

5. 潘锦平，《软件开发技术》，上海科技出版社，1985年

6. 何克清等，《基于面向对象计算模型的软件工程研究》，武汉大学出版社，1989．10

7. 冯玉琳、丁茂顺，"系统开发的时序模型方法"，《计算机研究与发展》，1989．1．

8. Ding Maoshun et al.,Interactive Environment Building System BUILDER. In "Adanvances In Chinese Computer Science", Vol. 1. 1988. Singapore

9. E. Yourdon L. Costontine. Structured desiggn. Prentice Hall. 1979.

10. T. Demarco, Structured Analysis and System Specification. 1979

11. M. A. Jackson, Principles of program design. Academic Press, 1976.

12. M. A. Jackson,System Develepment. 1983

13. A. Goldberg. D. Robson, Smalltalk-80. The Language and its Implementation. Addison-Wesely. 1983.

14. 中国软件行业协会，《系统分析员教材》，1989．

• 交换规则 (exchange)

$$\frac{A,\ C,\ D,\ A' \vdash B}{A,\ D,\ C,\ A' \vdash B}LX$$

说明如果从某些假设可以推出结论B，那么相同的结论也可以从这些假设的任何排列得到。这意味着推理与资源使用的顺序无关。

• 缩规则 (contraction)

$$\frac{A,\ C,\ C \vdash B}{A,\ C \vdash B}LC$$

即如果结论B可由同一公式C的两个假设导出，那么它也可由此公式的单一假设推出。这意味着任何假设一旦引入，则可重复使用。

• 弱规则 (weakening)

它使得某些空假设的使用成为可能，即允许一个不使用所有假设的推理。由于缩规则和弱规则，我们可以随心所欲地使用一个假设，这是古典逻辑中资源使用失控的主要原因。去除这两条规则，将得到一个线性系统，其中任一假设被使用且仅被使用一次。与之相应，我们必须对逻辑连接词进行某些修改。对应于古典逻辑，线性系统有两类连接词：加性连接词和乘性连接词。在乘性框架中不存在资源共享的问题，如规则 $\frac{\Gamma \vdash A,\ \Delta \vdash B}{\Gamma,\ \Delta \vdash A \otimes B}\otimes$；而在加性框架中，则需考虑共享，如规则 $\frac{\Gamma \vdash A,\ \Gamma \vdash B}{\Gamma \vdash A\&B}\&$。另外，线性逻辑还引入了模态词!(of course)，!A去除了对A的使用限制。从这个意义上说，直觉主义逻辑和传统逻辑都可以自然地借助于模态词解释到LL。重要的是模态词的使用是局部的，资源的任意使用仅限于那些标志为!的公式，整个逻辑框架仍是线性的。

对资源使用的这种考虑相应于线性λ-演算，其中任一抽象项 λx • M都有性质：x在M中自由出现次数为1。因此它也对应于任何过程被调用次数恰为1的一种程序设计语言。在LL中，最重要的是线性非(⊥)，它把所有连接词联系在一起。每个原子公式都以两种形式出现A及线性非 $A^{\perp}$ 满足 $A^{\perp\perp}=A$，注意⊥不是一种连接词，但它可以通过 de Morgan性质作用于复合公式，如

$$(A\otimes B)^{\perp}=A^{\perp}\parr B^{\perp} \qquad (A\ \&B)^{\perp}=A^{\perp}\oplus B^{\perp} \qquad (!A)^{\perp}=?A^{\perp}$$

$$(A\parr B)^{\perp}=A^{\perp}\otimes B^{\perp} \qquad (A\oplus B)^{\perp}=A^{\perp}\ \&B^{\perp} \qquad (?A)^{\perp}=!A^{\perp}$$

在LL的矢列演算中，由于LL的对称性，我们可以把矢列演算中的左边公式移到右边，如公理A⊢A改写成$\vdash A^{\perp}, A$，因此仅需考虑矢列右边的演算。下面给出LL的矢列演算：

公理 $\vdash A^{\perp}, A$

cut $\dfrac{\vdash C, A \quad \vdash C^{\perp}, B}{\vdash A, B}cut$

结构规则 $\dfrac{\vdash A, C, D, B}{\vdash A, D, C, B}X$

逻辑规则 $\dfrac{\vdash C, A \quad \vdash D, B}{\vdash C\otimes D, A, B}\otimes$ $\dfrac{\vdash C, D, A}{\vdash C\parr D, A}\parr$

$\dfrac{}{\vdash 1}1$ $\dfrac{\vdash A}{\vdash \perp, A}\perp$ $\dfrac{\vdash C, A \quad \vdash D, A}{\vdash C\&D, A}\&$

$\dfrac{\vdash C, A}{\vdash C\oplus D, A}1\oplus$ $\dfrac{\vdash D, A}{\vdash C\oplus D, A}2\oplus$

$\dfrac{}{\vdash T, A}T$ $\dfrac{\vdash A}{\vdash ?B, A}W?$

$\dfrac{\vdash ?B, ?B, A}{\vdash ?B, A}C?$ $\dfrac{\vdash B, ?A}{\vdash !B, ?A}!$

$\dfrac{\vdash B, A}{\vdash ?B, A}D?$

这里1，⊥，T，0分别是⊙，⅋，&，⊕的单位元，有性质$1^{\perp}=\perp$，$0^{\perp}=T$，$\perp^{\perp}=1$，$T^{\perp}=0$

2. 证明网

2.1. 从矢列演算到证明网

矢列演算存在许多冗余，如规则 $\dfrac{\vdash C, D, A, B}{\vdash C\parr D, A, B}\parr$ 中，A未被改变。如果全部去除这些冗余，我们就得到证明网。一个矢列演算对应一个唯一的证明网，但每个证明网至少对应一个矢列演算。下面我们给出证明网的构造规则：

• link $\overbrace{A \quad A^{\perp}}$ 是证明网；

• 如果A是证明网v的结论，$A^{\perp}$是证明网v'的结论，那么$\dfrac{\begin{matrix}v\\ \vdots\\ A\end{matrix}\quad\begin{matrix}v'\\ \vdots\\ A^{\perp}\end{matrix}}{cut}$是一证明网；

• 如果A和B是同一证明网v的结论

那么 $\dfrac{\begin{matrix}v\\ \vdots\\ A\end{matrix}\quad\begin{matrix}\\ \vdots\\ B\end{matrix}}{A\otimes B}$ 是一证明网；

• 如果A，B分别是证明网v，v'的结论

那么 $\dfrac{\begin{matrix}v\\ \vdots\\ A\end{matrix}\quad\begin{matrix}v'\\ \vdots\\ B\end{matrix}}{A\otimes B}$ 是一证明网。

2.2 证明网和自然推理

可以说证明网是LL的自然推理，但它和直觉主义的自然推理有两点不同

• 无需假设块及杀死假设块的discharge操作；

• 无需⇒的引进和消去规则。

如果从直觉主义隐含和线性隐含A⊸B的相似性出发，我们有：

$$\dfrac{\begin{matrix}[A]\\ \vdots\\ B\end{matrix}}{A\Rightarrow B}\Rightarrow I \quad \text{对应于} \quad \dfrac{\begin{matrix}|\\ A\\ \vdots\\ A^{\perp}\quad B\end{matrix}}{{}^{\perp}A\,\Re\,B}\ (A\multimap B)$$

$$\dfrac{\begin{matrix}\vdots\\ A\Rightarrow B\end{matrix}\quad\begin{matrix}\vdots\\ A\end{matrix}}{B}\Rightarrow e \quad \text{对应于} \quad \dfrac{\begin{matrix}\vdots\\ A\end{matrix}\quad B^{\perp}}{A\otimes B^{\perp}}\ cut\ B$$

直观地说，可以认为在证明网中，一个假设块A只含A的一次出现，并且一旦此假设块被引入，即被杀死。

2.3 证明网的范式化和cut消去

证明网给出了一个消去cut的明了图解，证明网的转换规则如下：

$$\overbrace{\begin{matrix}A\\ \vdots\end{matrix}\quad\begin{matrix}A^{\perp}\quad A\\ \vdots\end{matrix}}\ cut \quad\hookrightarrow\quad \begin{matrix}\vdots\\ A\end{matrix}$$

$$\dfrac{\begin{matrix}\vdots\quad\vdots\\ A\quad B\end{matrix}\quad\begin{matrix}\vdots\quad\vdots\\ A^{\perp}\quad B^{\perp}\end{matrix}}{A\otimes B\quad A^{\perp}\,\Re\,B^{\perp}}\ cut \quad\hookrightarrow\quad \dfrac{\begin{matrix}\vdots\quad\vdots\\ A\quad A^{\perp}\end{matrix}}{cut}\quad\dfrac{\begin{matrix}\vdots\quad\vdots\\ B\quad B^{\perp}\end{matrix}}{cut}$$

cut 的消去过程就叫证明网的范式化过程，有性质：

1)一个证明网由上述规则转换成一证明网；

2)cut的消去满足CR性质；

3)任何一个证明网均可归约到一无 cut 证明网。

重要的是cut的消去次序是任意的，亦即 cut 的消去可以并行进行。

2.4 线性非和假设、结论对称性

上面我们已看到，证明网是一个只有结论的证明结构。之所以仅需考虑结论，是因为线性非（⊥）具有改变假设/结论的功能。

在自然推理中，一个有假设H，结论C的推理可图示如下

$$\begin{matrix}H\\ \mathbb{D}\ \vdots\\ C\end{matrix}$$

若我们有一H的证明$\mathbb{D}'$，那么我们有

$$\dfrac{\begin{matrix}[H]\\ \mathbb{D}\ \vdots\\ C\end{matrix}}{H\Rightarrow C}\Rightarrow I \quad\hookrightarrow\quad \begin{matrix}\mathbb{D}'\ \vdots\\ H\\ \mathbb{D}\ \vdots\\ C\end{matrix}$$

$$\dfrac{\mathbb{D}'\quad H\quad H\Rightarrow C}{C}\Rightarrow e$$

$$\vdots$$

与$\mathbb{D}$对应我们有证明网$\mathbb{D}H^{\perp}C$，它可以看成在假设H下推出结论C，也可看成在假设$C^{\perp}$下推出结论$H^{\perp}$，即

$$\overbrace{\begin{matrix}H\\ \vdots\\ C\end{matrix}\quad H^{\perp}} \quad\text{或}\quad \overbrace{\begin{matrix}C^{\perp}\\ \vdots\\ H^{\perp}\end{matrix}\quad C}$$

由此可以看出，证明网中一结论C的线性非$C^{\perp}$可以看成是其它结论的一个假设，即⊥改变了假设/结论。

2.5 线性逻辑连接词的含义

上节我们已描述了线性非的作用，下面进一步观察LL中其他连接词的意义

2.5.1 **乘性连接词和合作关系** 若已

有类型为A，B的两个输出，那么可用 A⊗B 或A⅋B来表示A和B之间的某种合作关系，其中A⊙B表示A，B相互独立，无需通讯。A⅋B表示A和B互相依赖，存在通讯。一种典型的通讯是结论C对假设H的依赖关系，记成H⊥⅋C。

2.5.2 **加性连接词和共享** 这里共享指的是在类型为A，B的两个输出中，恰好一个是有效的。这有两种情况：

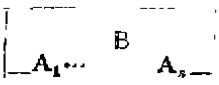
1) 开共享 A&B。指A和B两个输出都能被产生，但不知那个是有效的。一个典型的例子是if语句，它的两个选择项都已被考虑但不知那个是所需的。

2) 闭共享 A⊕B。只有A或B两者之一被产生。

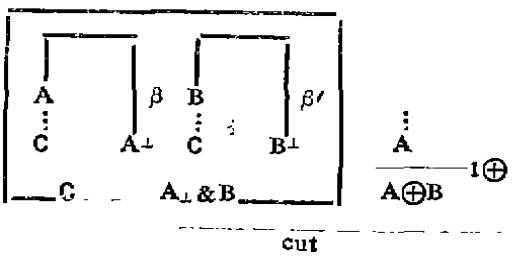
对应闭共享，我们在构造证明网时可使用规则：

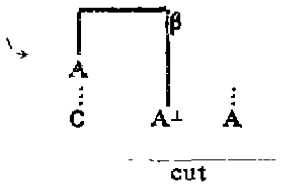
$$\frac{A}{A\oplus B}\,1\oplus \qquad 或 \qquad \frac{B}{A\oplus B}\,2\oplus$$

对应开共享，情况较复杂。我们需要引入Proof-box的概念，一个Proof-box是如下所示的一个带有若干结论的方框：



一个box可以通过规则和另外的box或公式相连，这种连接和box的内容无关，这样规则&可以描述如下，给定结论为C，A⊥和C，B⊥的两个证明网β，β′，我们形成一个结论为A⊥&B⊥和C的box，其中A⊥&B⊥称为主入口，C称为辅助入口，一个范式化的例子如下：





当然我们也可把box的辅助入口的公式和其它公式相连，但这可能产生许多无益的工作。因此，我们最好是等待主要入口处相应公式的出现。上例中A的出现相当于条件为真，这使我们仅需考虑证明网β的转换，而证明网β′整个消失了。这种情况也可称为同步，即我们构造一个box但不对box中的内容作进一步的转换，而是等待合适公式的出现。

2.5.3 **指数连接词和存取操作** 指数连接词和内存的使用有关，与类型为A的输入有关的指令记成!A或?A，分别表示下述存取操作：

1) 存贮 规则 $\dfrac{\vdash A,?B}{\vdash\,!A,?B}\,!$ 表示把一个输出A存放于内存并等待一个类型为A的内存维护指令的出现；

2) 内存维护

(1)复制 规则 $\dfrac{\vdash ?A,?A,B}{\vdash ?A,B}\,C?$ 表示内存中一个类型为A的单元的内容的复制；

(2)擦去 规则 $\dfrac{\vdash B}{\vdash ?A,B}\,W?$ 表示等待一个类型为A的存贮指令的出现，并将其所存内容擦去；

(3)读取 规则 $\dfrac{\vdash A,\ B}{\vdash ?A,B}\,D?$ 表示对内存中一个类型为A的单元的一次读取。

指数规则的使用使得在一个计算中，内存的使用成为可能。

2.6 Curry-Howard同构

意大利Solitro提出的ν演算作为与证明网之间的Curry-Howard同构[3]，其基本思想与赋类型λ-演算及自然推理之间的同构思想类似。证明网中每个原子公式对应一个不

同的赋类型变量，而类型就是证明网中的公式。按照证明网的结构，相应的ν项可以逐步构造出来。

ν演算有与证明网类似的性质，如ce性质，强范式化定理等。

3. 线性抽象机，LL的语义模型及范畴描述

3.1 线性抽象机LAM

在第一节我们谈到线性直觉主义逻辑，它是施加线性限制于直觉主义逻辑的结果。由于线性直觉主义逻辑是一种Heyting逻辑，因此我们有等价的范畴组合子表示（称为线性组合子）。矢列演算中cut的消去就对应于相应范畴组合子的计算，而范畴组合子的计算可在范畴抽象机（CAM）上进行，由于线性组合子的线性性质，Lafont[4]发现CAM可简化为线性抽象机LAM。它有下述特点

1）自然的Lazy和Strict计值　在CAM上考虑一类型为 $(U\to V)\wedge U$ 的对(f,x)，范畴组合子fst，snd，app都可作用其上，但fst，snd的计值是Lazy的，而app的计值是严格的，因此难以确定要进行何种计值。在LL中，我们有两种合取连接词，对应于⊗，不存在投影组合子 $A\otimes B\to A$ 或 $A\otimes B\to B$，它要求严格计值；而对应于&，其上的操作只能是fst，snd，因此它的计值是Lazy的。进一步的分析表明连接词⅋，⊕也是Lazy计值的。

2）无需不用单元收集器（Garbage Collector）　由于&是Lazy计值的，⊗不存在投影和配对组合子，因此环境以树的形式出现，它既不存在共享结点也不存在废弃结点，当然无需回收废弃单元。

3）副作用　由于不存在共享，也就没有副作用问题。

由上我们看到LL揭示了函数式语言（FP）实现的一个光辉前景。事实上如果不考虑效率可以实现任何函数式语言到LAM的编译。

3.2 语义模型

Girard给出了LL的两个语义模型

·阶段语义（Phase semantics），其基本思想是把公式解释成事实，把连接词解释为事实上的运算，它对于线性谓词演算是有效且完全的[1]。

·相关语义（Coherent semantics），它是在系统F的语义模型上发展起来的，有兴趣的读者可参看〔1〕,〔5〕,〔6〕。

3.3 范畴描述

我们知道自然推理中的证明等价于赋类型的λ-演算，而后者可以用笛卡儿闭范畴（CCC）来刻划。有趣的是，早在1977年Barr就给出了LL对应的范畴描述，这就是*-自治范畴。事实上LL对应的范畴就是具某些特性的对称类群闭范畴（SMC）。读者可参阅〔7〕,〔8〕,〔9〕。

4. 目前发展和结束语

自87年Girard的文章发表以来，LL已引起了理论计算机学家和数学工作者的广泛注意。他们在各自的领域中，用LL作为描述、分析和解决问题的工具，取得了诱人的成就，我们看到

·在函数式语言的实现上，除了LAM外，进一步的工作还证实LL可应用于优化工作[10]；

·在逻辑式程序设计语言的设计中，也提出LL作为一种控制[11]；

·在并发性描述上，Oliet等人分析了LL和Petri网的相似性后指出LL可能作为一种并发系统的规范，并在其上发展一种并发程序设计语言[12]；

·Lafont最近的研究也说明了LL可以用来作为描述连接机（connection machine）计算行为的一种模型[13]。

LL不仅把许多不同研究领域联系在一起，并且开拓了各自新的研究方向。当然作为一个新的逻辑系统，LL还有许多工作有待我们去做，但我们已看到它给证明论及计算机科学带来的曙光。（转第41页）

Ch. 7, Reasoning About Reasoning, by D, B, Lenat, R. Davis, J. Doyle, M. Genesereth, I. Goldstein, and H. Schrobe

2. L. Aiello, C. Cecchi, and D. Sartini, Repre-Sentation and Use of Metaknowledge, PIEEE Vol. 74, No, 10, Oct, 1986
3. E. Sandewall, Nonmonotonic Inference Rules for Multiple Inheritance with Exceptions, PIEEE Vol. 74, No, 10, Oct. 1986
4. A. Yonezawa and T. Watenabe, An Introduction to Object-Based Reflective Concurrent Computation, Proc. of the ACM SIGPLAN Workshop on Object-Based Concurrent Programming, SAN DIEGO, Sept. 26-27, 1988
5. P. Maes, Concepts and Experiments in Computational Reflection, ACM SIGPLAN Notices, Vol. 22, No, 12, Dec. 1987
6. J. Ferber, Computational Reflection in Class Based Object Oriented Languages, OOPSLA'89 Proceedings, Oct. 1—6, 1989
7. P. Cointe, A Tutorial Introduction to Metaclass Architecture as Provided by Class Oriented Languages, Proc. of the International Conf. on 5th Generation Computer Systems, Nov. 28—Dec. 2, 1988, Tokyo, Vol. 2, 592—603
8. N. Graube, Metaclass Compatibility, OOPSLA '89 Proceedings, Oct. 1—6, 1989
9. 陈平、蔡希尧、金益民, An Approach to Introduce the Reflection to C++, Proc. COMPSAC'90, Oct. 1990
10. 陈平、蔡希尧、金益民, Object Identity is Object, A Reflective Approach in C++, Submitted to Toulouse'90
11. B. Stroustrup, The C++ Programming Language, Addison-Wesley Pub., 1986

(接第19页)

参考文献

[1] J. Y. Girard. Linear Logic. TCS, 50: 1-102, 1987.
[2] J. Y. Girard. Y. Lafont. Linear logic and lazy computation. In, Springer LNCS 250, 1987.52-66.
[3] U. Solitro. A typed calculus based on a fragment of linear logic. TCS, 68: 333-342, 1989
[4] Y. Lafont. The linear abstract machine. TCS, 59: 157-180, 1988
[5] J. Y. Girard, et al. Proofs and types. Cambrige Tracts in Theoretical Computer Science, Cambridge Universiry Press, 1989
[6] J. Y. Girard. The System F of variable types, fifteen years later. TCS, 45: 159-192, 1986
[7] R. A. G. Sleey. Linear logic, *-autonomous categories and cofree coalgebras. Contemporary Mathematics, Vol. 92, 1989. 371-382
[8] J. Y. Girard. Toward a geometry of interaction. Contemporary Math Vol. 92, 1989
[9] V. C. V. de Paviva. A Dialectica-like model of linear logic. In LNCS 389. 1989.341—356.
[10] J. C. Guzman, et al. Single-threaded polymorphic λ-calculus. In Proc. 5-th IEEE Symp. on Logic in Computer Science, Philadelphia, June 1990.
[11] S. Cerrito. A linear semamtics for allowed logic programs. In Proc. 5-th IEEE Symn. on Logic in Computer Science, Philadelphia, June 1990.
[12] N. Marti-Oliet et al. From Petri nets to linear logic. IN LNCS 389, 1989. 313-340
[13] Y. Lafont. Interaction nets. In Proc. 17-th ACM Symp. on Principles of Programming Languages, San Francisco, 95-108, Jan. 1990.
[14] J. Y. Girard, Linear logic and parallelism, in: Proc. School on Semantics of Parallelism, IAC (CNR, Roma, 1986)