項書き換えシステム入門

外山 芳人

北陸先端科学技術大学院大学 情報科学研究科
923-1292 石川県辰口町旭台 1-1
e-mail: toyama@jaist.ac.jp

項書き換えシステムは等式にもとづく柔軟な計算法と効率的な証明法を提供できるため、定理自動証明、関数型あるいは論理型言語、代数的仕様記述、記号処理など、計算機科学のさまざまな分野で広くもちいられている。ここでは、項書き換えシステムの理論において中心的な役割をはたす合流性について紹介し、自動証明において項書き換えシステムがどのように応用されているかを解説する。

項書き換えシステム、合流性、自動証明、完備化手続き

Introduction to Term Rewriting Systems

Yoshihito Toyama

School of Information Science, JAIST
Tatsunokuchi, Ishikawa 923-1292, Japan
e-mail: toyama@jaist.ac.jp

Term rewriting systems can offer both flexible computing and effective reasoning with equations, and have been widely applied to automated theorem proving, functional and logic programming, algebraic specification, and symbolic computation. In this paper we first introduce the confluence property, which is certainly one of the most fundamental properties of term rewriting theory, and present applications of term rewriting systems to the field of automated theorem proving.

term rewriting systems, confluence property, automated theorem proving, completion

1 はじめに

項書き換えシステム (term rewriting system) は等式にもとづく柔軟な計算法と効率的な証明法を提供できるため、定理自動証明、関数型あるいは論理型言語、代数的仕様記述、記号処理など、計算機科学のさまざまな分野で広くもちいられている。

項書き換えシステムは方向付けられた等式（書き換え規則）の集合として定義される。ところで、等式そのものにはもともと計算という意味はない。たとえば、等式 $1+2=3$ は右辺と左辺が等価であるという意味をもつだけであり、$1+2$ から 3 を得るばかりではなく、逆に 3 から $1+2$ を得ることも可能である。しかし、この等式を計算という立場でながめてみると、$1+2$ の計算結果として 3 が得られるのであり、3 の計算結果として $1+2$ が得られることはない。このように、等式 $1+2=3$ を複雑な式から単純な式への非可逆な書き換え規則 $1+2 \to 3$ とみなすと、等式の世界を計算の世界に自然な形で結び付けることが可能となる。

項書き換えシステムの計算は、これらの書き換え規則を繰り返し適用することによって与えられた項がもっとも単純な形（正規形）に到達するまでリダクションすることで実現される。したがって、項書き換えシステムをもちいることにより、等式にもとづいて記述された関数型プログラムや代数的仕様記述に、自然な形でリダクションにもとづく操作的意味を与えることができる。また、自動証明における等式推論をリダクションで実行することにより、証明を効率的な計算に置き換えることも可能となる。

項書き換えシステムの理論は、論理と計算の二つの世界にまたがるさまざまな問題に対し、数学的によく整理された統一的な枠組を提供している。ここでは、項書き換えシステムの理論において中心的な役割をはたすリダクション過程の合流性（チャーチ・ロッサ性）について紹介し、自動証明において項書き換えシステムがどのように応用されているかをわかりやすく解説する。

2 項書き換えシステム

2.1 再帰プログラムの計算

再帰プログラムの計算を例にとり項書き換えシステムの考え方を説明する。等式で記述された再帰プログラムの例として階乗関数 $f(n)=n!$ を考えよう。

$$f(n) = \text{if zero}(n) \text{ then } 0 \text{ else } n * f(n-1) \qquad (1)$$

ここで zero(n) は n の値が 0 のときのみ true、それ以外では false となる論理式である。このとき $f(3)$ は再帰プログラムによって以下のように計算される。

$$\begin{aligned} f(3) &\to \text{if zero}(3) \text{ then } 1 \text{ else } 3 * f(3-1) \\ &\to 3 * f(2) \\ &\to 3 * (\text{if zero}(2) \text{ then } 1 \text{ else } 2 * f(2-1)) \\ &\to \cdots \to 3 * (2 * (1 * 1)) \to 6 \end{aligned}$$

この計算において等式 (1) は左辺から右辺への書き換え規則としてもちいられていることに注意する。すなわち、与えられた項 $f(3)$ を最終的な答えに徐々に近づけて行く非可逆な計算過程を自然に表現するためには、等式よりも左辺から右辺への書き換え規則 (2) の方が自然である。

$$f(n) \to \text{if zero}(n) \text{ then } 1 \text{ else } n * f(n-1) \qquad (2)$$

このように、システムを記述している等式 $s=t$ を左辺から右辺への書き換え規則 $s \to t$ とみなしたものが項書き換えシステム (term rewriting systme) である。

2.2 項書き換えシステムの例

前節の再帰プログラムでは、 $*$、 $-$、zero、if 文の意味は自明なものとして自由に計算をおこなった。しかし、これらもすべて書き換え規則で表すことができる。たとえば、自然数上の加算 $+$ を項書き換えシステムで表してみよう。話を簡単にするために、以下では 0、 1、 2、 $\cdots$ を 0、$s(0)$、$s(s(0))$、$\cdots$ で表現することにする。すると自然数 x の次の数は $s(x)$ となるから加算 $+$ は以下の等式システム E_+ で形式化できる。

$$E_+ \quad \begin{cases} x+0=x \\ x+s(y)=s(x+y) \end{cases}$$

ここで E_+ の等式を左辺から右辺への書き換えシステムとみなすと項書き換えシステム R_+ を得ることができる。

$$R_+ \quad \begin{cases} x+0 \to x \\ x+s(y) \to s(x+y) \end{cases}$$

このとき $1+2=3$ の計算は R_+ をもちいて以下のリダクションを行うことによって得られる。

$$s(0)+s(s(0)) \to s(s(0)+s(0)) \to s(s(s(0)+0)) \to s(s(s(0))).$$

ここで 項 $s(s(s(0)))$ はこれ以上リダクションすることができない。このような項を正規形 (normal form) という。正規形はリダクションによる計算の答えとみなすことができる。項 s から 項 t (t は正規形である必要はない) に 0 回以上のリダクションで到達でき

るとき $s \xrightarrow{*} t$ と書く。したがって上記のリダクションは $s(0)+s(s(0)) \xrightarrow{*} s(s(s(0)))$ と表せる。

加算以外の関数も同様に書き換え規則であらわすことにより、階乗関数 f を定める項書き換えシステム R_f は以下のように得られる。

$$
R_f \left\{ \begin{array}{l}
x+0 \to x \\
x+s(y) \to s(x+y) \\
x-0 \to x \\
s(x)+s(y) \to x-y \\
x*0 \to 0 \\
x*s(y) \to (x*y)+x \\
\text{if true then } x \text{ else } y \to x \\
\text{if false then } x \text{ else } y \ \to y \\
\text{zero}(0) \to \text{true} \\
\text{zero}(s(x)) \to \text{false} \\
f(x) \to \text{if zero}(x) \text{ then } x \text{ else } x*f(x-1)
\end{array} \right.
$$

ここで与えた項書き換えシステム R_f ではすべての計算が書き換え規則のみで完全に実現されていることに注意されたい。このように項書き換えシステムではすべての計算を書き換え規則によって定める。実際、すべての計算可能な関数は有限個の書き換え規則で完全に表せることが知られており、項書き換えシステムは計算モデルとして十分な計算能力をもっている。

3　項書き換えシステムの合流性

ここでは、項書き換えシステムの基本的な性質である合流性 (チャーチ・ロッサ性) について簡単に紹介する。

項書き換えシステムでは、与えられた項からのリダクションの道筋は一通りとは限らない。項に適用可能な書き換え規則も、その規則を適用できる部分も一般には一意に決まらないからである。したがって、書き換え規則の適用順序の違いによって、何通りものリダクションの道筋が可能となる。たとえば、階乗を計算する R_f における項 $(s(0)*0)+(s(0)+0)$ のリダクションは図 1 のようになる。

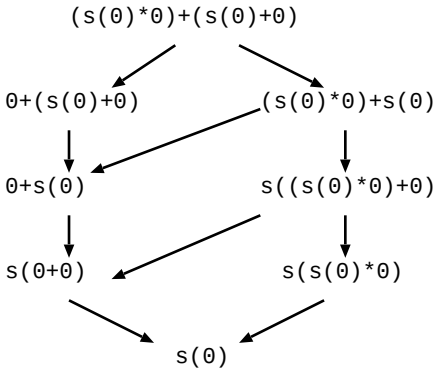


図 1. R_f **のリダクション**

この例では途中のリダクションの道筋とは関係なく、すべてのリダクションは唯一の正規形 $s(0)$ に合流している。一般の項書き換えシステムでは、このような合流性は必ずしも保証されていない。もし、図 1 のようにリダクションの合流性が常に保証されているならば、項書き換えシステムは合流性 (チャーチ・ロッサ性) をみたすという。正確に述べると、項書き換えシステムが合流性をみたすとは、任意の項 t をリダクションして項 s、r を得たならば、必ず s と r から合流できる項 p が存在することである (図 2)。

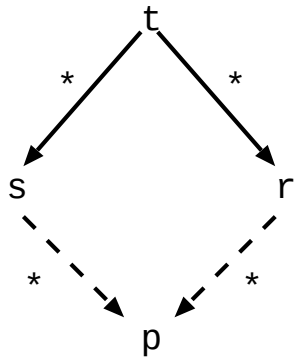


図 2. 合流性

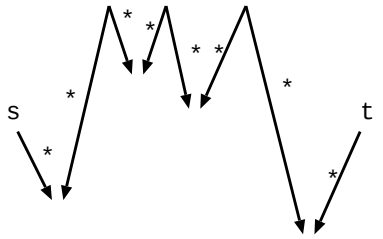
項書き換えシステム R が合流性をみたすならば次の重要な性質がなりたつ。

性質 1. 項 s の正規形は高々 1 個しか存在しない。

性質 2. 等式 $s=t$ が成立するならば、適当な項 r が存在して $s \xrightarrow{*} r$ かつ $t \xrightarrow{*} r$ となる。

ここで等式 $s=t$ が成立するとは、項書き換えシステム R を両方向への書き換え可能な等式システム E とみなしたとき、$s=t$ が E で証明できることである。 いいかえると、右から左あるいは左から右へのリダクションを適当に組み合わせることにより、s と t をジグザグにつなぐことが可能となることである。

ここで、先ほど述べた性質 1 と 2 の意味を考えてみよう。性質 1 は、s の正規形 t が存在するなら、どのようなリダクションで正規形を求めても t に一致



することを保証している。つまり、非決定的な計算において、どのような計算で答えを求めても常に正しい答えが得られることを意味している。これは、さまざまな操作的意味を可能とする関数・論理型言語のような計算システムの枠組においては、合流性がみたされている場合には、遅延計算、部分計算、並列計算などの柔軟な計算メカニズムを自由に駆使できることを意味している。

性質 2 は、等式システムによる $s = t$ の証明がリダクションによる効率的な計算でおこなえることを意味している。すなわち、通常の証明が試行錯誤を繰り返しながら進むのに対して、合流性をみたす項書き換えシステムによる証明では s と t のリダクションのみを考えれば十分であり、とくにリダクションによって正規形が必ず求まる場合には試行錯誤はまったく不要になる。このことを少し詳しく説明しよう。

項書き換えシステムのリダクションが必ず停止するならば、項書き換えシステムは停止性 (強正規性) をみたすという。項書き換えシステムが合流性と停止性をみたすなら、完備 (complete) であるという。すると性質 2 より、項書き換えシステム R が完備であるなら $s = t$ の証明は次のようにおこなえばよいことがわかる。まず、 s と t にリダクションをおこない 正規形 $s\downarrow$、$t\downarrow$ を求める。ここで、R の停止性からそれぞれのリダクションは必ず止まり正規形が求まることに注意する。このとき、二つの項が同じ形をしていることを $\equiv$ で表すと、$s\downarrow \equiv t\downarrow$ ならば $s = t$ は成立し、$s\downarrow \not\equiv t\downarrow$ ならば $s = t$ は成立しないことがわかる。つまり、完備な項書き換えシステムが与えられると、等式 $s = t$ の証明問題は決定可能となる。

このように、与えられた等式システム E を項書き換えシステム R とみなしたときに、R が完備になっているならば等式証明問題は効率的に解くことができる。しかし、一般には等式を左から右への書き換え規則とみなすだけでは、完備な項書き換えシステムは得られない。そこで、等式システム E を論理的に等価なシステムに変形することにより、完備な書き換えシステムを構成する手続きが広く利用されている。次に、このような完備化手続きについて説明しよう。

4　項書き換えシステムの完備化

まず、以下のパズルをもちいて完備化手続きの基本的なアイデアを説明する。

4.1　グラス置き換えパズル

酒のグラス S、ウィスキーのグラス W、ビールのグラス B が以下のように一列に並んでいる。

図 3

このとき、次のグラスの置き換えシステム G によって、グラス列を置き換えることが許されるものとする。

$$G \quad \begin{cases} W \leftrightarrow SW \\ WB \leftrightarrow S \end{cases}$$

置き換え規則 $W \leftrightarrow SW$ は W を 2 個のグラスの並び SW に置き換えてもよいし、その逆に SW を 1 個のグラス W に置き換えてもよいことを示している。たとえば、最初に示したグラス列 SWB は以下のようにグラス列 WBWB に置き換えることができる。

$SWB \leftrightarrow SSWB \leftrightarrow WBSWB \leftrightarrow WBWB$

このときグラス列 SWB と WBWB は置き換え規則 G によってつなぐことができるという。それでは以下のグラス列はうまくつながるだろうか。

問題

(1) $SSWB \leftrightarrow \cdots \leftrightarrow WBWB$?

(2) $SSSW \leftrightarrow \cdots \leftrightarrow WBWB$?

この問題の難しさは、二つのグラス列をつなぐことが不可能な場合に、どうやって不可能であることを示すかという点にある。もし、グラス列をつなぐことが可能ならば、試行錯誤を繰り返して答えを発見すればよい。しかし、不可能な場合には、すべての置き換えがグラス列をつなぐことに失敗することを示す必要がある。置き換え方法は無限にあるから、これらをすべて検査していては永久に答えは得られ

ない。そこで、完備な書き換えシステムをもちいてこのパズルを解く方法を考えてみよう。

4.2 グラス置き換えパズルの完備化

グラス置き換え規則を方向付けてやることにより、G から停止性をもつ書き換えシステム R を以下のように構成してみよう。

$$R \quad \begin{cases} SW \to W \\ WB \to S \end{cases}$$

ここで、書き換えシステム R が完備となるならば前節で説明したように、二つのグラス列がつながるか否かという問題は、グラス列の正規形が一致するか否かという問題に還元できるので完全に解くことが可能となる。書き換えシステム R のリダクションは書き換えによってグラス列を短くして行くので、R が停止性をみたすことは明らかである。しかし、R は合流性をみたさない。グラス列 SWB が以下のように二つの正規形 S と SS をもつからである。

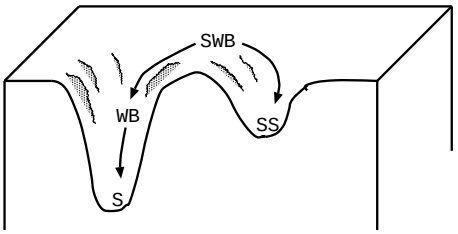


図 4

それでは、R がグラス列 SWB に対して合流性をみたさない原因を考えてみよう。図４のリダクションでは二つの書き換え規則 $SW \to W$ と $WB \to S$ の適用が SWB で可能となっており、グラス W でそれぞれの書き換え規則の左辺は重なってる。したがって、一方の書き換え規則の適用は、そこに重なっている他方の書き換え規則の適用を不可能にしてしまい、その結果としてリダクションの合流性が破壊されてしまう。この例では、二つの規則の左辺の重なり SWB にそれぞれの規則を適用して WB と SS が得られ、さらに WB と SS が異なる正規形をもつことから合流性が壊されている。

以上のように、二つの書き換え規則に重なりがあると合流性は無条件に成立しないことがわかる。二つの規則の左辺の重なり SWB に対して、それぞれの規則を一回づつ適用して得られる値の対 $< WB, SS >$ を危険対という。危険対の要素をそれぞれリダクションして正規形を求めたとき、両者が一致するなら危険対は収束する、一致しないならば発散するという。つまり、R が合流しないのは危険対 $< WB, SS >$ が発散しているからである。二つの書き換え規則の重なりは存在したとしても有限であるから、項書き換えシステムが有限個の書き換え規則しかもたないならば、その危険対も有限であることに注意する。ここで、以下の定理が知られている。

定理 (Knuth-Bendix の合流条件). 停止性をみたす項書き換えシステム R が合流性 (つまり完備性) をもつための必要十分条件は、R のすべての危険対が収束することである。

したがって、書き換えシステム R を完備にするためには、論理的に等価な変形をおこなって、発散する危険対をうまく消してやればよい。発散する危険対 $< WB, SS >$ は正規形 S と SS をもつ。このとき、これを一つの正規形に収束させるために、新しい書き換え規則 $SS \to S$ を 付け加えて、書き換えシステム R' をつくってみよう。

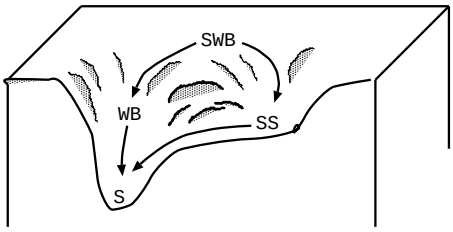


図 5

書き換えシステム R' は以下のようになる。

$$R' \quad \begin{cases} SW \to W \\ WB \to S \\ SS \to S \end{cases}$$

ここで注意してほしいのは、新しく付け加えられた規則 $SS \to S$ は、もとの置き換え規則を組み合わせることによって以下のように G でシミュレーションできることことである。

$SS \leftrightarrow SWB \leftrightarrow WB \leftrightarrow S$

したがって、グラス置き換えパズルの本質は、付け加えられた規則によって変更されていない。つまり、書き換えシステム R のかわりに書き換えシステム R' のもとでパズルを解いてもよいことがわかる。ここで、新しく得られた R' の書き換え規則の重なりをチェックすると、すべての危険対が収束することを容易に示すことができる。したがって、R' は完備な

書き換えシステムとなっている。

先ほどのグラス置き換え問題 (1) と (2) は、完備な R' によるリダクションで簡単に解くことができる。問題 (1) は $SSWB\downarrow \equiv S \equiv WBWB\downarrow$ となるから、グラス列は置き換え規則 G でつなぐことができる。一方、問題 (2) は、$SSSW\downarrow \equiv W \neq S \equiv WBWB\downarrow$ となるから、グラス列は G でつなぐことができないことがわかる。

4.3 Knuth-Bendix の完備化手続き

グラス置き換えパズルの例では、書き換えシステム R の発散する危険対から作られた書き換え規則を付け加えるだけで、完備な書き換えシステム R' を得ることができた。しかし、一般には新しい書き換え規則を付け加えることによって、発散する危険対が新たに生ずる可能性がある。したがって、このような場合には発散する危険対が完全になくなるまで、次々と新しい書き換え規則を付け加えていく必要がある。また、書き換え規則の両辺が他の規則によってリダクションできるのならば、両辺をリダクションして正規形にした方がシステムは単純になり、さらに発散する危険対の生成も押えられる。

このような点まで考慮した完備化手続きとして、有名な Kunuth-Bendix 完備化手続きを以下に示す。

Knuth-Bendix 完備化手続き

R は書き換え規則の有限集合、E は等式の有限集合、$>$ は停止性を保証する項上の順序とする。また、危険対 $\langle p, q\rangle$ を E の中では等式 $p = q$ で表す。

E が空でないならば以下を繰り返せ。E が空になれば完備化は**成功**。

(1) E から $s > t$ をみたす 等式 $s = t$ (あるいは $t = s$) をひとつ取り除く。もし、そのような等式がないなら完備化は**失敗**。

(2) 書き換え規則 $s \to t$ でリダクションできる R の書き換え規則をすべて E に移す。

(3) $s \to t$ と R によって生じたすべての危険対を E に付け加える。

(4) $s \to t$ を R に付け加える。

(5) R をもちいて R のすべての書き換え規則の右辺を正規形にする。

(6) R をもちいて E の等式の両辺をすべて正規形にする。両辺が一致した等式は E から取り除く。

ここで $>$ は停止性を保証するための適当な順序であり、R のすべての書き換え規則 $s \to t$ が $s > t$ をみたすなら R は停止性をみたすものとする。手続きは最初 R を空集合とし、適当な等式の集合 E を与えて実行を開始する。そして、E が空集合になれば完備化は成功し、完備な項書き換えシステム R が得られる (図 6)。なお、この手続きはループを永久に回り続けて停止しない場合があることに注意する。

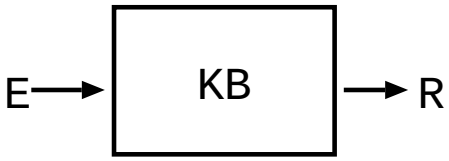


図 6

4.4 群の完備化

完備化手続きをもちいて群を完備化した例を示そう。群の公理である左単位元の存在、左逆元の存在、結合律を等式システム E_{group} で表す。

$$E_{group} \quad \begin{cases} 1 * x = x \\ I(x) * x = 1 \\ (x * y) * z = x * (y * z) \end{cases}$$

ここで、完備化手続きを E_{group} に適用すると、最終的には以下の 10 個の規則からなる完備な項書き換えシステム R_{group} を得ることができる

$$R_{group} \quad \begin{cases} I(1) \to 1 \\ 1 * x \to x \\ x * 1 \to x \\ I(I(x)) \to x \\ I(x) * x \to 1 \\ x * I(x) \to 1 \\ I(x * y) \to I(y) * I(x) \\ (x * y) * z \to x * (y * z) \\ I(x) * (x * y) \to y \\ x * (I(x) * y) \to y \end{cases}$$

したがって、E_{group} のもとで等式 $s = t$ が成立するか否かという語の問題は、R_{group} によって完全に解くことができる。なぜなら、すでに説明したように、E_{group} のもとで等式 $s = t$ が成立するか否かという問題は、R_{group} のもとで $s\downarrow$ と $t\downarrow$ が一致するか否かという問題に還元されるからである。

4.5 リスト反転の完備化

次にリストの反転関数 $reverse$ の完備化例を示す。リスト上の関数 $append$ と $reverse$ を 等式システム E_{rev} で表す。E_{rev} は反転関数の代数的仕様とみなすこともできる。

$$E_{rev} \left\{ \begin{array}{l} append(nil, y) = y \\ append(cons(x, y), z) = \\ \qquad cons(x, append(y, z)) \\ reverse(nil) = nil \\ reverse(cons(x, y)) = \\ \quad append(reverse(y), cons(x, nil)) \\ reverse(reverse(x)) = x \end{array} \right.$$

ここで等式システム E_{rev} を完備化すると、完備な項書き換えシステム R_{rev} が得られる。

$$R_{rev} \left\{ \begin{array}{l} append(nil, y) \to y \\ append(cons(x, y), z) \to \\ \qquad cons(x, append(y, z)) \\ reverse(nil) \to nil \\ reverse(reverse(x)) \to x \\ reverse(cons(x, y)) \to \\ \quad append(reverse(y), cons(x, nil)) \\ reverse(append(x, cons(y, nil))) \to \\ \qquad cons(y, reverse(x)) \end{array} \right.$$

R_{rev} をもちいると、群の語の問題と同様に、リスト上の等式が成立するか否かを決定することができる。また、R_{rev} のリダクションを計算と考えると、これは仕様 E_{rev} から生成されたプログラムとみなすことも可能である。このとき、R_{rev} の完備性から、計算の道筋に無関係に答は一意に定まることが保証されている。

5 おわりに

項書き換えシステムの重要な性質である合流性と、その自動証明への応用であるク Knuth-Bendix 完備化手続きについて説明した。1980 年代になって本格的に開始された項書き換えシステムの研究は、今日では計算機科学の重要な研究分野のひとつとして広く認められ、多くの研究成果が蓄積されつつある。したがって、ここで紹介した話は項書き換えシステムの理論と応用のごく一部に過ぎない。項書き換えシステムについてさらに知りたい読者には、参考書として Baader-Nipkow (1998) を推薦する。これは、英語で書かれた最初の本格的な教科書で、取り上げている話題のバランスの良さ、記述レベルの適切さ、豊富な演習問題など、この分野の全貌を勉強するのに現時点では最適な 1 冊といえる。

ソフトウェア・サイエンスとの関連で考えると、プログラミング言語の研究において操作的意味論のはたす役割は近年ますます増大している。それにともない、その理論的基礎である項書き換えシステムに対する関心も年々高まっている。項書き換えシステムやラムダ計算などのリダクション・システムが関数型プログラミング言語の意味論の中ではたす役割については Mitchell (1996) が詳しい。

参考文献

[1] F. Baader and T. Nipkow, Term rewriting and all that (Cambridge University Press, 1998).

[2] J. C. Mitchell, Foundations for programming languages (MIT Press, 1996).