Controlled Grammars

Informal Definition

Label	Condition	Kernel	F_T	F_F
l	Q(, ,)	A(, ,) —> B(, ,)	L_T	L_F

Parameters (variables and functions) are shown in parenthesis.

If condition Q is true, production with label *l* is applied and we go to the production with label from L_T.

If Q is not true, production *l* does not apply and we go to the production from L_F.

Values of parameters are changed when we apply productions.

Controlled Grammars

A controlled grammar **G** is the following eight-tuple:

$$\mathbf{G} = (V_T, V_N, V_{PR}, E, H, Parm, L, R),$$

where

V_T is the alphabet of *terminal symbols*;

V_N is the alphabet of *nonterminal symbols*, *S* (from V_N) is the start symbol;

V_{PR} is the alphabet of the *first order predicate calculus PR*:

$V_{PR} = Truth \cup Con \cup Var \cup Func \cup Pred \cup \{\text{symbols of logical operations}\}$,

where

Truth are truth symbols T and F (these are reserved symbols);

Con are constant symbols;

Var are variable symbols;

Func are functional symbols ($Func = Fcon \cup Fvar$). Functions have an attached non-negative integer referred to the *arity* indicating the number of elements of the domain mapped onto each element of the range. A term is either a constant, variable or function expression.

A *function expression* is given by a functional symbol of arity *k*, followed by *k* terms, $t_1, t_2, \ldots, t_k$, enclosed in parentheses and separated by commas;

Pred are predicate symbols. Predicates have an associated positive integer referred to as *arity* or "argument number" for the predicate. Predicates with the same name but different arities are considered distinct. An *atom* is a predicate constant of arity *n*, followed by *n* terms, $t_1, t_2, \ldots, t_n$, enclosed in parentheses and separated by commas. The truth values, *T* and *F*, are also atoms. *Well-formed formulas* (or WFF) are atoms and combinations of atoms using logical operations;

H is an *interpretation* of *PR* calculus on the set *E*,

Parm is a mapping from $V_T \cup V_N$ in 2^{Var} matching with each symbol of the alphabet $V_T \cup V_N$ a set of formal parameters, with $Parm(S) = Var$;

L is a finite set called the set of *labels*;

R is a finite set of *productions*, i.e., a finite set of the following seven-tuples:

$$(l, Q, A \longrightarrow B, \pi_k, \pi_n, F_T, F_F).$$

Controlled Grammars

(l, Q, A—>B, π_k, π_n, F_T, F_F).

Here

l (from L) is the label of a production; the labels of different productions are different, and subsequently sets of labels will be made identical to the sets of productions labeled by them;

Q is a WFF of the predicate calculus *PR* , the *condition* of applicability of productions; Q contains only variables from *Var* which belong to *Parm*(A);

A—>B is an expression called the *kernel of production*, where

A is from V_N;

B is from $(V_T \cup V_N)^*$ is a string in the alphabet of the grammar G;

π_k is a sequence of functional formulas corresponding to all formal parameters of each entry of symbols from $V_T \cup V_N$ into the strings A and B (*kernel actual parameters*);

π_n is a sequence of functional formulas corresponding to all formal parameters of each functional symbol from *Fvar* (*non-kernel actual parameters*);

F_T is a subset of L of labels of the productions permitted on the next step of derivation if Q = T (“true”); it is called a *permissible set in case of success*;

F_F is a subset of L of labels of the productions permitted on the next step of derivation if Q = F (“false”); it is called a *permissible set in case of failure*.

Structure of a typical controlled grammar

L	Q	Kernel, π_k	π_n	F_T	F_F
l_i	Q_i	$A(, ,)$ —> $B(, ,)$			

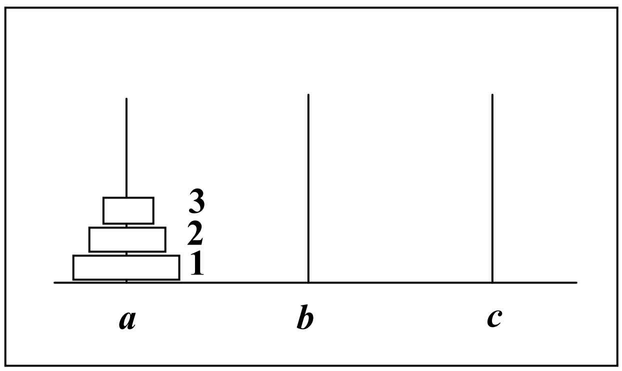
V_T = ... **V_N = ...** **V_{PR} = ...**

E is ... ***Parm*: ...**

Controlled Grammars

The Tower of Hanoi Problem

The problem is as follows. There are three pivots ***a***, ***b***, and ***c***. On the first one there is a set of ***n*** disks, each of different radius. The task is to move all the disks to the pivot ***c*** moving only one disk at a time. In addition, at no time during the process may a disk be placed on top of a smaller disk. The pivot ***c*** can, of course, be used as a temporary resting place for the disks.



Let us designate an elementary step of moving disk number i from the pivot x to the pivot y as ***p***(i, x, y), a terminal symbol with parameters. Thus a solution of the Tower of Hanoi Problem might be represented as the following string of symbols with parameters:

$$\boldsymbol{p}(i_1, x_1, y_1)\boldsymbol{p}(i_2, x_2, y_2)...\boldsymbol{p}(i_m, x_m, y_m).$$

This is the string of the language of all possible sequences of moves. Consider the controlled grammar shown in Figure 4. We will apply this grammar for derivation of a solution for the case of three disks: n =3, x = ***a***, y = ***c***. It means that the values of parameters for the starting symbol ***S*** are ***S***(3, ***a***, c).

Controlled Grammars

Controlled grammar generating solutions to the Tower of Hanoi Problem

L	Q	Kernel, π_k	π_n	F_T	F_F
1	Q_1	$S(n, x, y) \to A(n, x, y)$		2	ø
2	Q_2	$A(n, x, y) \to A(f_1(n), x, f_2(x, y))$ $p(n, x, y)$ $A(f_1(n), f_2(x, y), y)$		2	3
3	Q_3	$A(n, x, y) \to p(n, x, y)$		2	ø

Here $V_T = \{p\}$

$V_N = \{S, A\}$

V_{PR}

Pred = $\{Q_1, Q_2, Q_3\}$,

$Q_1 = T$

$Q_2(n) = T$, if $n > 1$; $Q_2(n) = F$, if $n = 1$.

$Q_3(n) = T$, if $n = 1$; $Q_3(n) = F$, if $n > 1$.

Var = $\{n, x, y\}$

$F = Fcon \cup Fvar$,

Fcon = $\{f_1, f_2\}$

$f_1(n) = n - 1$, $n = 2, 3, \ldots$

$f_2(x, y)$ yields the value from $\{a, b, c\} - \{x, y\}$,

where values of

x, y are from $\{a, b, c\}$

Fvar = $\{3, a, c\}$

$E = Z_+ \cup \{a, b, c\}$

Parm: $S \to Var$, $A \to Var$, $p \to Var$

$L = \{1, 2, 3\}$

At the beginning of derivation: $x = a$, $y = c$, $n = 3$.

Controlled Grammars

Derivation of a solution in case of n = 3:

S(3, ***a***, ***b***) 1=> ***A***(3, ***a***, ***c***)2=> ***A***(2, ***a***, ***b***)***p***(3, ***a***, ***c***)***A***(2, ***b***, ***c***)
2=> ***A***(1, ***a***, ***c***)***p***(2, ***a***, ***b***)***A***(1, ***c***, ***b***)***p***(3, ***a***, ***c***)***A***(2, ***b***, ***c***)
3=> ***p***(1, ***a***, ***c***)***p***(2. ***a***, ***b***)***A***(1, ***c***, ***b***)***p***(3, ***a***, ***c***)***A***(2, ***b***, ***c***)
3=> ***p***(1, ***a***, ***c***)***p***(2, ***a***, ***b***)***p***(1, ***c***, ***b***)***p***(3, ***a***, ***c***)***A***(2, ***b***, ***c***)
2=> ***p***(1, ***a***, ***c***)***p***(2, ***a***, ***b***)***p***(1, ***c***, ***b***)***p***(3, ***a***, ***c***)***A***(1, ***b***, ***a***)
p(2, ***b***, ***c***)***A***(1, ***a***, ***c***)
3=> ***p***(1, ***a***, ***c***)***p***(2, ***a***, ***b***)***p***(1, ***c***, ***b***)***p***(3, ***a***, ***c***)***p***(1, ***b***, ***a***)
p(2, ***b***, ***c***)***A***(1, ***a***, ***c***)
3=> ***p***(1,***a***, ***c***)***p***(2,***a***, ***b***)***p***(1, ***c***, ***b***)***p***(3, ***a***, ***c***)***p***(1, ***b***, ***a***)
p(2, ***b***, ***c***)***p***(1, ***a***, ***c***).

Controlled Grammars

Derivation of a solution in case of n = 4:

S(4, *a*, *c*) ^{1}=> *A*(4, *a*, *c*)^{2}=> <u>*A*(3, *a*, *b*)</u>*p*(4, *a*, *c*)*A*(3, *b*, *c*)

<u>*A*(3, *a*, *b*)</u>^{2}=> *A*(2, *a*, *c*)*p*(3, *a*, *b*)*A*(2, *c*, *b*)
^{2}=> *A*(1, *a*, *b*)*p*(2, *a*, *c*)*A*(1, *b*, *c*)*p*(3, *a*, *b*)*A*(2, *c*, *b*)
^{3}=> *p*(1, *a*, *b*)*p*(2, *a*, *c*)*A*(1, *b*, *c*)*p*(3, *a*, *b*)*A*(2, *c*, *b*)
^{3}=> *p*(1, *a*, *b*)*p*(2, *a*, *c*)*p*(1, *b*, *c*)*p*(3, *a*, *b*)*A*(2, *c*, *b*)
^{2}=>*p*(1, *a*, *b*)*p*(2, *a*, *c*)*p*(1, *b*, *c*)*p*(3, *a*, *b*)*A*(1, *c*, *a*)*p*(2, *c*, *b*)*A*(1, *a*, *b*)
^{3}=> *p*(1, *a*, *b*)*p*(2, *a*, *c*)*p*(1, *b*, *c*)*p*(3, *a*, *b*)*p*(1, *c*, *a*)*p*(2, *c*, *b*)*A*(1, *a*, *b*)
^{3}=> *p*(1, *a*, *b*)*p*(2, *a*, *c*)*p*(1, *b*, *c*)*p*(3, *a*, *b*)*p*(1, *c*, *a*)*p*(2, *c*, *b*)*p*(1, *a*, *b*).

$^{2,..,3}$=> *p*(1,*a*, *b*)*p*(2,*a*, *c*)*p*(1, *b*, *c*)*p*(3, *a*, *b*)*p*(1, *c*, *a*)*p*(2, *c*, *b*)*p*(1, *a*, *b*)
p(4, *a*, *c*)<u>*A*(3, *b*, *c*)</u>

<u>*A*(3, *b*, *c*)</u>^{2}=> *A*(2, *b*, *a*)*p*(3, *b*, *c*)*A*(2, *a*, *c*)
^{2}=> *A*(1, *b*, *c*)*p*(2, *b*, *a*)*A*(1, *c*, *a*)*p*(3, *b*, *c*)*A*(2, *a*, *c*)
^{3}=> *p*(1, *b*, *c*)*p*(2. *b*, *a*)*A*(1, *c*, *a*)*p*(3, *b*, *c*)*A*(2, *a*, *c*)
^{3}=> *p*(1, *b*, *c*)*p*(2, *b*, *a*)*p*(1, *c*, *a*)*p*(3, *b*, *c*)*A*(2, *a*, *c*)
^{2}=>*p*(1, *b*, *c*)*p*(2, *b*, *a*)*p*(1, *c*, *a*)*p*(3, *b*, *c*)*A*(1, *a*, *b*)*p*(2, *a*, *c*)*A*(1, *b*, *c*)
^{3}=> *p*(1, *b*, *c*)*p*(2, *b*, *a*)*p*(1, *c*, *a*)*p*(3, *b*, *c*)*p*(1, *a*, *b*)*p*(2, *a*, *c*)*A*(1, *b*, *c*)
^{3}=> *p*(1, *b*, *c*)*p*(2, *b*, *a*)*p*(1, *c*, *a*)*p*(3, *b*, *c*)*p*(1, *a*, *b*)*p*(2, *a*, *c*)*p*(1, *b*, *c*).

$^{2,..,3}$=> *p*(1,*a*, *b*)*p*(2,*a*, *c*)*p*(1, *b*, *c*)*p*(3, *a*, *b*)*p*(1, *c*, *a*)*p*(2, *c*, *b*)*p*(1, *a*, *b*)
p(4, *a*, *c*)
p(1, *b*, *c*)*p*(2, *b*, *a*)*p*(1, *c*, *a*)*p*(3, *b*, *c*)*p*(1, *a*, *b*)*p*(2, *a*, *c*)*p*(1, *b*, *c*).

Mathematical Induction

Let's prove by induction that 1+2+3+...+n = n(n+1)/2

---

1. For n=1 this is true: 1=1(1+1)/2

2. Assume that the statement is true for n = m, i.e.,
1+2+3+...+m = m(m+1)/2

3. Let's prove that the statement is true for n=m+1.
Indeed, from the assumption 2. we have
1+2+3+...+m + (m+1) = m(m+1)/2 + (m+1) =
(m/2+1) (m+1) = [(m+2)/2](m+1) = (m+1)(m+2)/2.
The statement is proved for n= m+1.

4. By induction we conclude that this statement is true for all n = 1, 2, 3, ...

Consider the following problem:
1+3+5+7+ ... + (2n – 1) = ?
Find the answer and prove by induction.

Solution

1. **$1 = 1^2$**

2. **Assume that $1 + 3 + 5 + ... + (2m - 1) = m^2$**

3. **$1+2+3+...+(2(m+1) - 1) = 1+2+3+...+ (2m -1) + (2m+1) = m^2 + (2m +1) =$ $m^2 + 2m +1 = (m +1)^2$**

Consider grammar $G = (V_T, V_N, P, S)$, $V_T = \{a, b\}$; $V_N = \{S\}$.

(1) **S —> aSb,**

(2) **S —> ab**

Prove that $\mathbf{a^n b^n}$ is the only language, which can be generated by this grammar.

1. We can easily check that our atement is true for k = 1, i.e., all the strings of this language of the length 2 = 2*1 are as follows: **ab.**
2. Assume that our statement is true for $k \leq n$, i.e., all the strings of this language of the length $2k \leq 2n$ are as follows: $\mathbf{a^k b^k}$.
3. Let us prove that all the strings of this language of the length 2k = 2(n+1) are as follows: $\mathbf{a^{n+1} b^{n+1}}$.

Consider an arbitrary string Z of this language of length 2(n+1). Let us reconstruct derivation of this string by going backward from the last step. In the course of derivation of this string the very last production applied has been production (2) because final string does not include non-terminal symbol S. Thus, the last step has been:

(*) $$X_n S Y_n \overset{2}{\longrightarrow} X_n \mathbf{ab} Y_n \quad (= Z)$$

where length $|X_n Y_n| = 2n$. Obviously, $X_n Y_n$ may include only terminals because productions (1) and (2) may generate strings, which include only one nonterminal symbol S or non of them. The only production that allows us to keep nonterminal S in the string is production (1). Thus, the previous step has been

(**) $$X_{n-1} S Y_{n-1} \overset{1}{\longrightarrow} X_{n-1} \mathbf{aSb} Y_{n-1},$$

where

(***) $$X_{n-1}\mathbf{a} = X_n, \ \mathbf{b}Y_{n-1} = Y_n$$

and the length $|X_{n-1} Y_{n-1}| = 2(n-1)$.

Instead of applying (**), let us apply production (2) to the same string:

$$X_{n-1} S Y_{n-1} \overset{2}{\longrightarrow} X_{n-1} \mathbf{ab} Y_{n-1}.$$

Thus, $X_{n-1}\mathbf{ab}Y_{n-1}$ is a string of terminal symbols and its length $|X_{n-1}\mathbf{ab}Y_{n-1}| = 2n$. From the assumption of induction 2. we conclude that this string must look like $X_{n-1}\mathbf{ab}Y_{n-1} = \mathbf{a^n b^n}$ and, consequently, $X_{n-1} = \mathbf{a^{n-1}}$, $Y_{n-1} = \mathbf{b^{n-1}}$. From (***) we conclude that $X_n = \mathbf{a^n}$, $Y_n = \mathbf{b^n}$. Thus, from (*) $Z = X_n \mathbf{ab} Y_n = \mathbf{a^{n+1} b^{n+1}}$.

4. By the principle of mathematical induction we conclude that this statement is true for all k = 1, 2, 3, ...

Induction for the Tower of Hanoi problem

Let's prove that the grammar generates a solution of this problem in general case. We shall prove by induction.

1. We proved that the grammar generates a solution of this problem for n = 3.

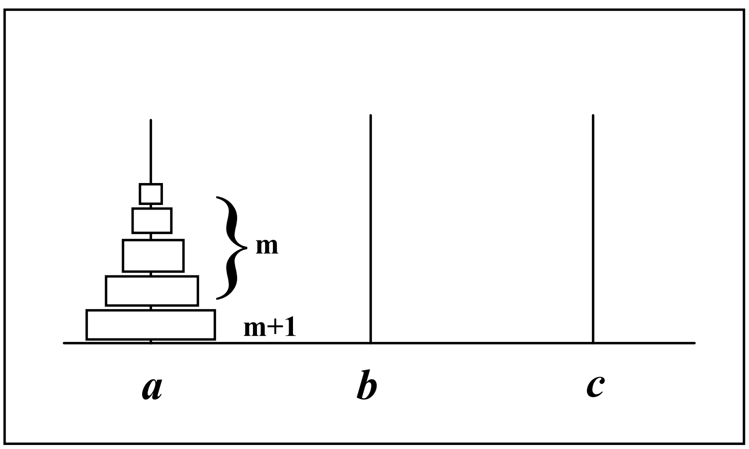
2. Assume that the grammar generates a solution for n = m.

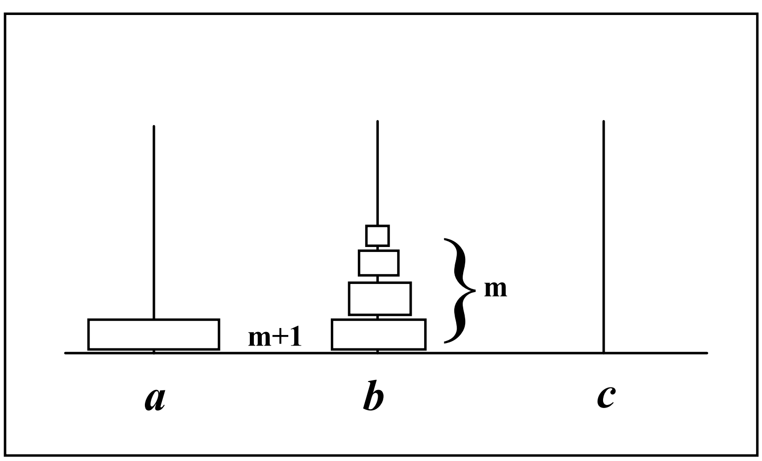
3. Let's prove that it generates a solution for n = m+1

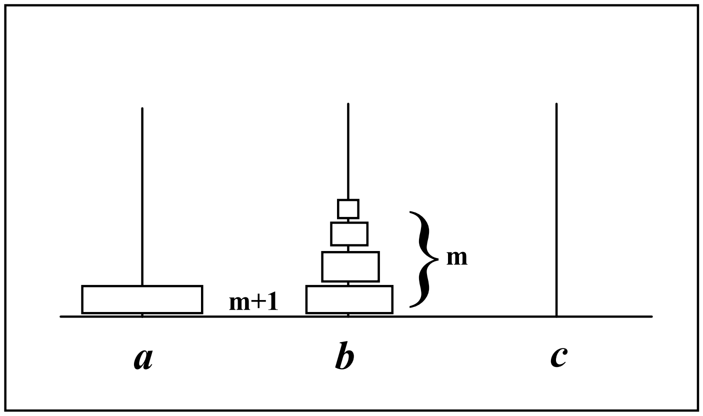
Consider the derivation in case of n = m+1:

***S*(m+1, *a*, *c*) ^{1}=> *A*(m+1, *a*, *c*) ^{2}=> *A*(m, *a*, *b*)*p*(m+1, *a*, *c*)*A*(m, *b*, *c*).**

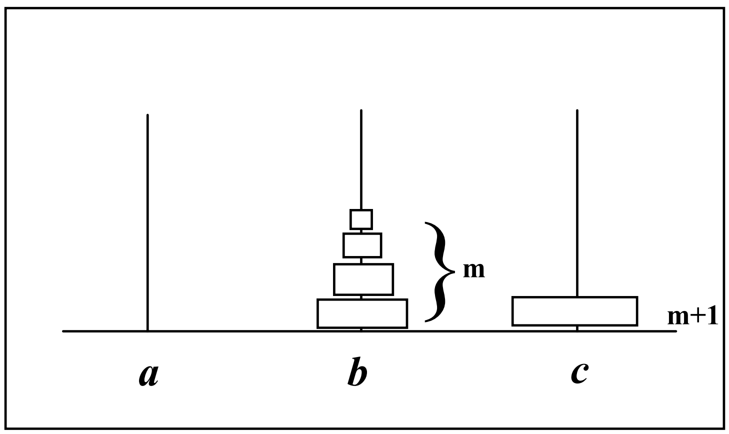
Obviously, the subsequent application of the grammar to the symbol *A*(m, *a*, *b*) will generate the string of symbols. According to the assumption of induction (2) this string corresponds to the solution of the Tower of Hanoi problem with m disks on the pivot *a*. These disks must be moved to the pivot *b*.





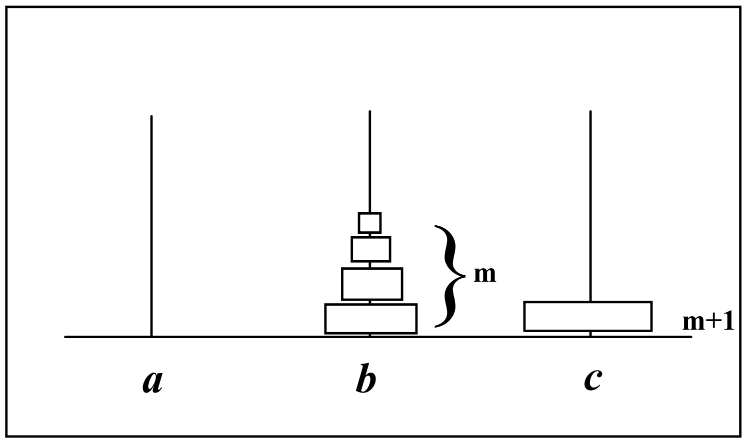


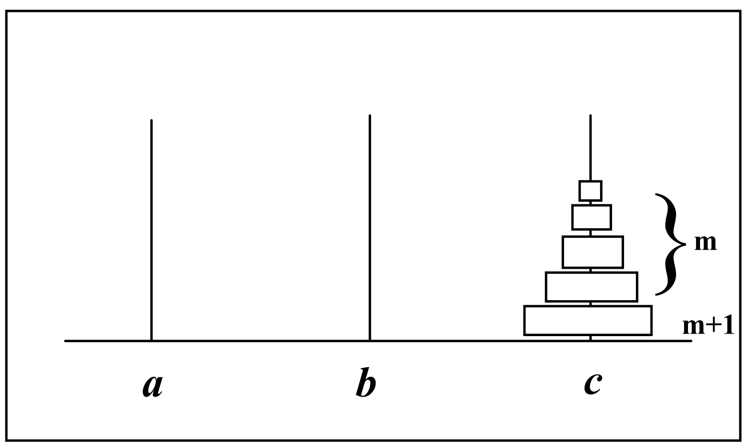
When these disks are moved to the pivot b we can apply *p*(m+1, *a*, *c*), i.e., we can move disk m+1 from pivot *a* to pivot *c*.



S(**m+1,** ***a*****,** ***c*****)** ^{1}**=>** ***A*****(m+1,** ***a*****,** ***c*****)** ^{2}**=>** ***A*****(m,** ***a*****,** ***b*****)*****p*****(m+1,** ***a*****,** ***c*****)*****A*****(m,** ***b*****,** ***c*****).**

The subsequent application of the grammar to the symbol *A*(m, *b*, *c*) will generate the string of symbols. According to the assumption of induction 2 this string corresponds to the solution of the Tower of Hanoi problem with m disks on the pivot *b*. These disks must be moved to the pivot *c*.





It means that the grammar generates a solution for n = m+1.

4. Conclusion: by induction the grammar generates a solution of the Tower of Hanoi problem for all n = 1, 2, 3, ...

More details about controlled grammars can be found in the book: Stilman, B., "Linguistic Geometry: From Search to Construction", Kluwer Academic Publishers (now Springer), 2000.