

Structures with $P = NP$ with Respect to the Uniform Model of Computation



The uniform model of computation:

- The size of an input (x_1, \dots, x_n) : n .
- The cost of the execution of one instruction: 1.

Concepts and Notations

$$\Sigma = (S; \underbrace{c_1, \dots, c_k}_{\text{constants}}; \underbrace{g_1, \dots, g_{k'}}_{\text{operations}}; \underbrace{r_1, \dots, r_{k''}}_{\text{relations}}, =)$$

Σ is a structure of finite signature, $k \geq 2$.

$$A_1, A_2 \subseteq S^\infty = \bigcup_{n=1}^{\infty} S^n$$

A_1, A_2 are problems.

$$f : S^\infty \mapsto S^\infty \text{ with } f(x) \in A_2 \Leftrightarrow x \in A_1$$

f is a function reducing A_1 to A_2 .

$$A_1 \leq_{pol} A_2, \quad A_1 \rightarrow_{[pol]} A_2$$

A_1 can be reduced to A_2 by a function in polynomial time.

$$A_1 \xrightarrow[f]{pol} A_2$$

The function f reduces A_1 to A_2 in polynomial time.

$$A \xrightarrow[R]{\text{dec}} \{c_1\}$$

A is decidable by means of R in constant time.

$$\forall (A \in \text{NP}) (A \leq_{pol} A_0)$$

A_0 is NP-hard.

SAT

The Satisfiability Problem.

Conditions for P = NP

We have P = NP for a structure if we can show one of the following reductions for each problem $A \in \text{NP}$.

$$(1) \quad A \xrightarrow[f_1]{pol} SAT \xrightarrow[f_2]{pol} \{c_1\}$$

The idea to define a new relation goes back to B. Poizat.

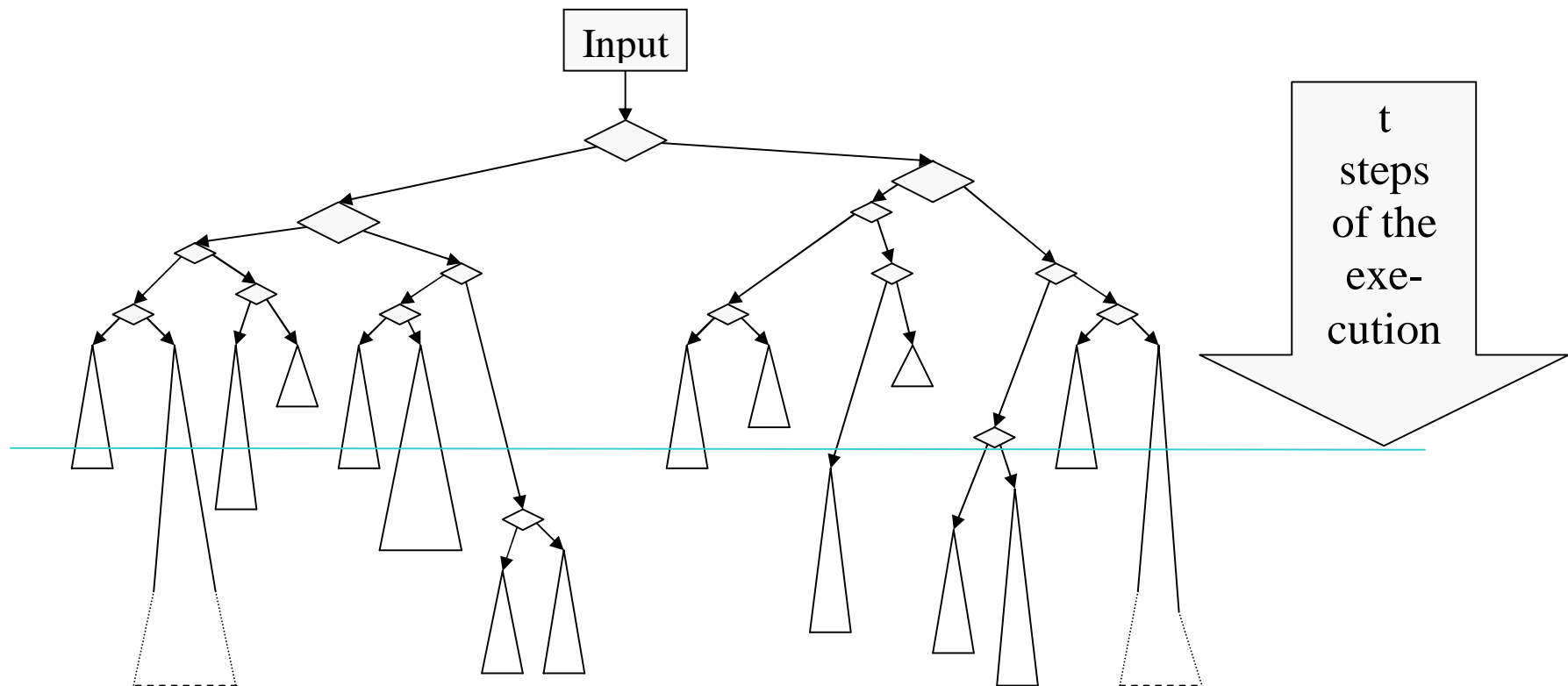
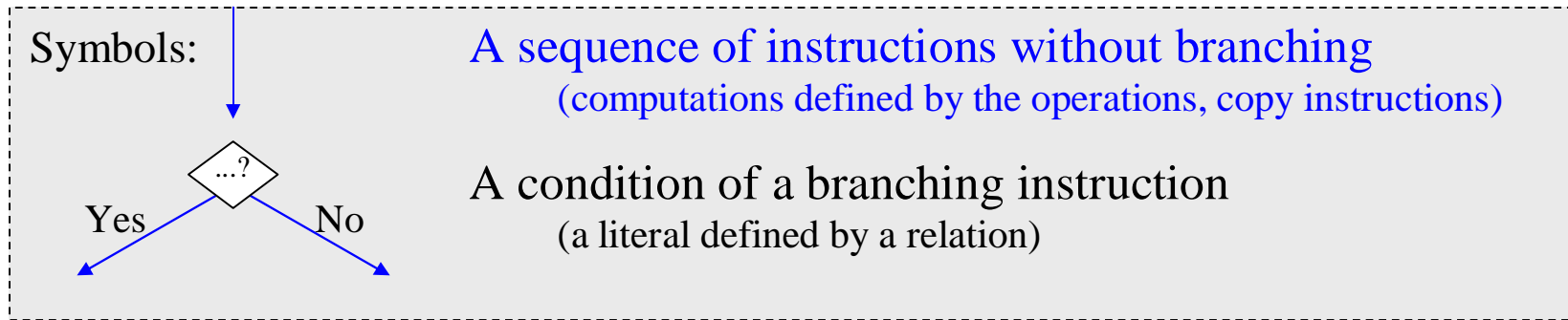
or

$$(2) \quad \underbrace{A}_{\subseteq S^\infty} \xrightarrow[f_1]{pol} \underbrace{SAT}_{\subseteq S^\infty} \xrightarrow[f_2]{pol} \underbrace{A_0}_{\subseteq S} \xrightarrow{R} \underbrace{\{c_1\}}_{\subseteq S}$$

Tuples are encoded by means of single elements.

and so on.

What about a Tree of Computation Paths?

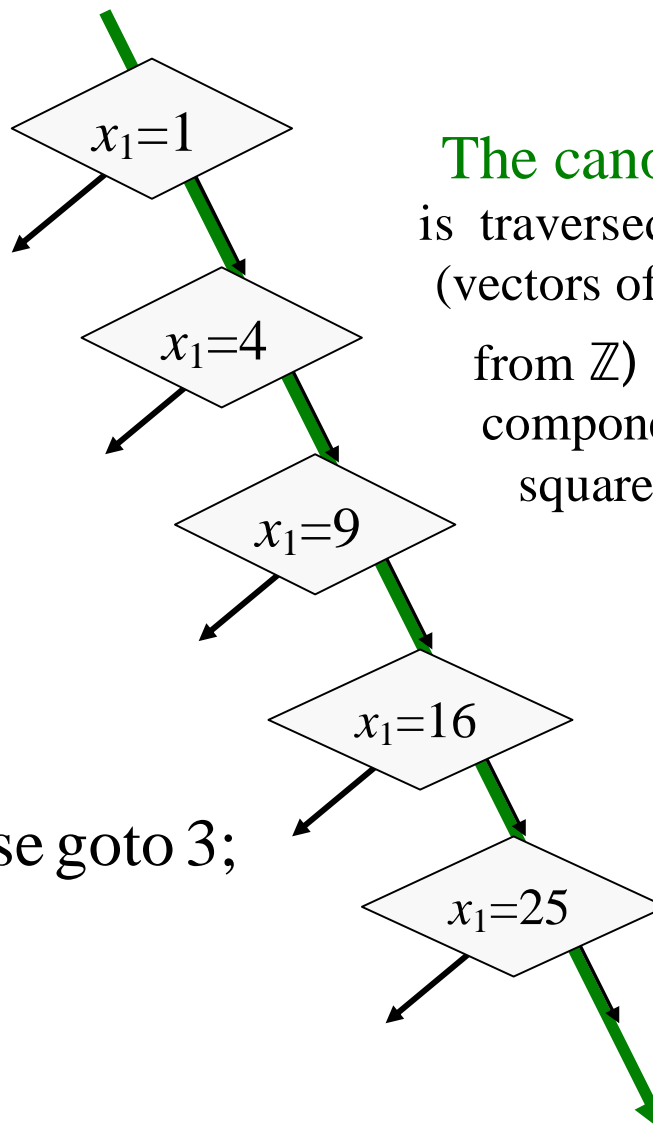


What about the Canonical Path? – An Example (A)

A structure: $(\mathbb{Z}; 0, 1; +, \cdot; =)$.

A program of a machine:

- 1: Input (x_1, \dots, x_n) ;
size: n
- 2: $x_2 := 0$;
- 3: $x_2 := x_2 + 1$;
- 4: $x_3 := x_2 * x_2$;
- 5: if $x_1 = x_3$ then goto 6 else goto 3;
- 6: $x_1 := 1$;
- 7: Output x_1 .



The canonical path is traversed by all tuples (vectors of components from \mathbb{Z}) whose first component is not a square number.

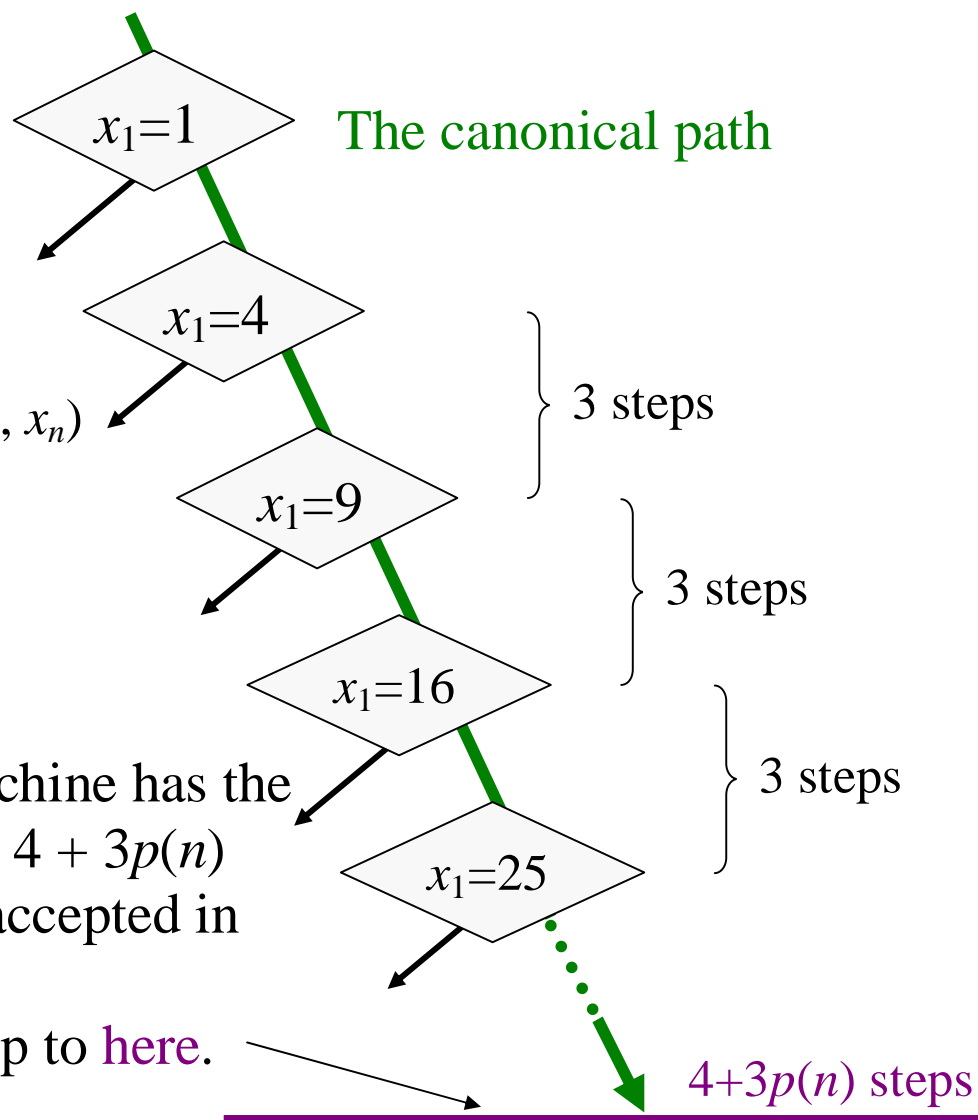
What about the Canonical Path? – An Example (B)

All inputs accepted by the machine only traverse a finite initial segment of the canonical path.

- The n -tuples $(1, x_2, \dots, x_n)$, $(4, x_2, \dots, x_n)$, \dots , $((p(n))^2, x_2, \dots, x_n)$ are accepted by the machine in $4 + 3p(n)$ steps.

- ! For *all the other* inputs, the machine has the *same* behaviour during the first $4 + 3p(n)$ steps. These inputs can not be accepted in this time.

There is no more information up to **here**.



A Structure of Paths Denoted by Strings and the Reduction

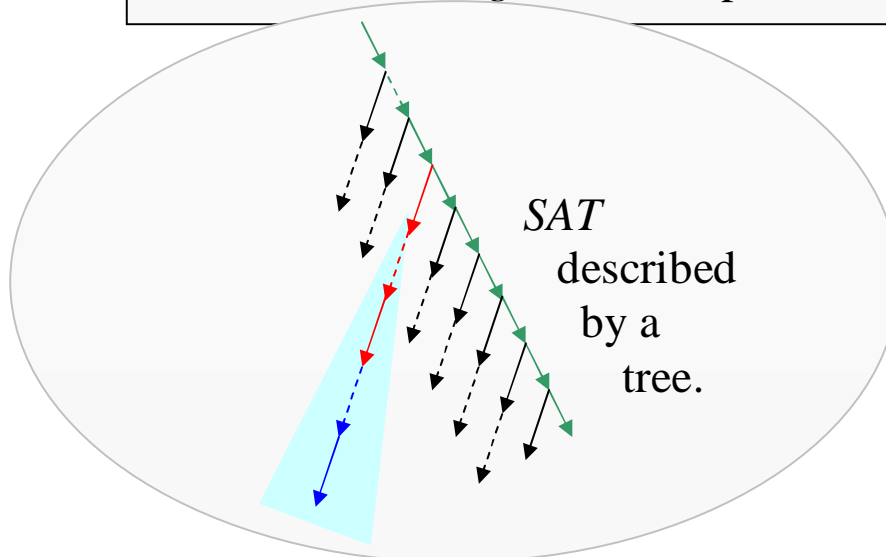
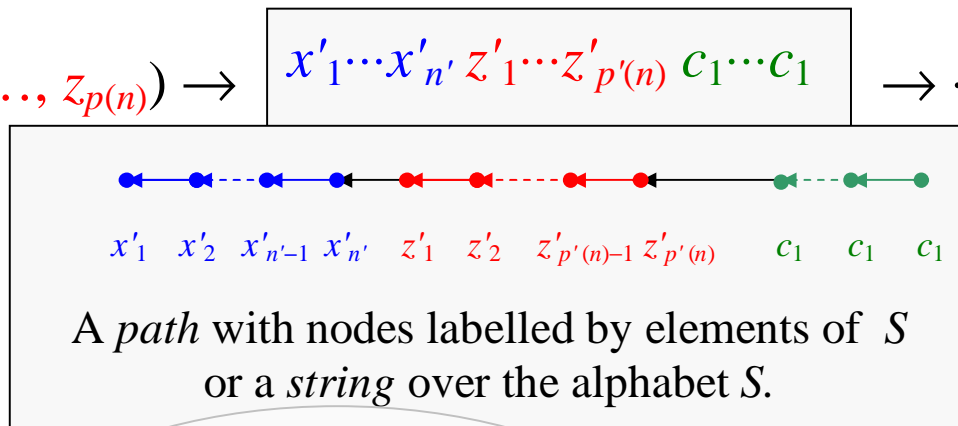
$$\underbrace{A}_{\subseteq (S^*)^\infty} \rightarrow \underbrace{SAT}_{\subseteq (S^*)^\infty} \rightarrow \underbrace{A_0}_{\subseteq S^*} \xrightarrow{R} \underbrace{\{c_1\}}_{\subseteq S}$$

Particularly,

$$(x_1, \dots, x_n) \in S^n \rightarrow (x_1, \dots, x_n, z_1, \dots, z_{p(n)}) \rightarrow \boxed{x'_1 \cdots x'_{n'} z'_1 \cdots z'_{p'(n)} c_1 \cdots c_1} \rightarrow \dots$$

$A \in NP$.

Let $(z_1, \dots, z_{p(n)}) \in \{c_1, c_2\}^{p(n)}$ be the code of a formula which is true for any tuple from $(S^*)^n$ iff this tuple is accepted by the NP-machine recognized A non-deterministically. (We use unary codes for the indices.)



Problems for which We Need a Solution

- The tuples have to be encoded by means of single elements.
 - Extension of the structure.
We shall use paths (denoted by strings) as elements.

- We need a new operation for the concatenation of paths or strings.
 - Expansion of the structure.

- We need a new relation.
 - Expansion of the structure.

Problems in Defining R and a Solution

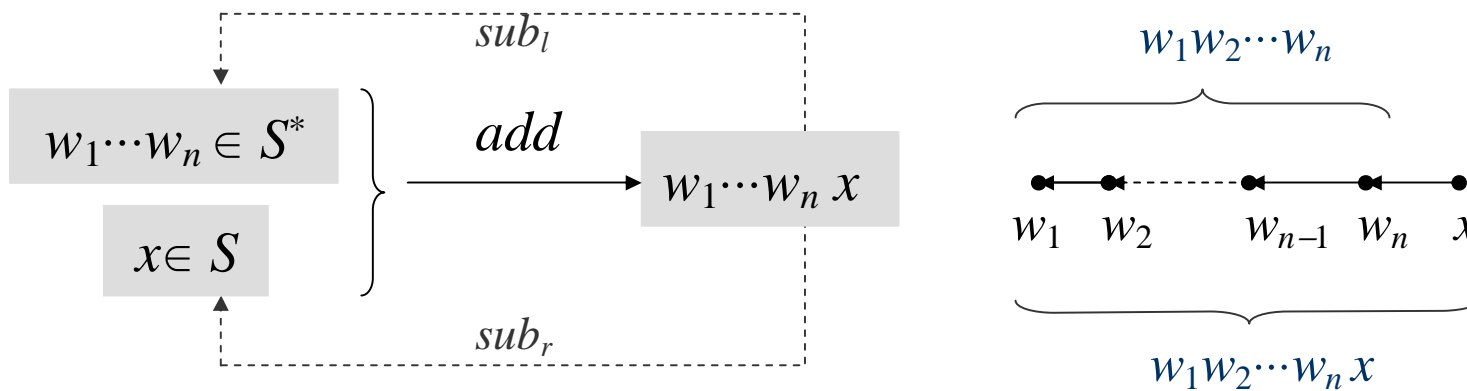
- For the new structure: SAT is decidable $\Leftrightarrow R$ is definable.
 - The different evaluations of the free variables have to be *finitely* describable.

We shall replace arbitrary paths by *short* paths.
(As the replacements of numbers in a paper by P. Koiran.)
- The formulae of SAT contain R .
 - *Recursive* definition of R .
- The formulae of SAT also contain free variables.
 - If there is an evaluation of the variables satisfying a formula from SAT , then we also want to have an evaluation of these variables by paths which are *shorter than the code of the corresponding tuple* from SAT (in general).

We use operations allowing *few modifications* of paths only such that we can restrict the free variables in the formulae with “=” to paths of polynomial length.

⇒ The *new* structure:

$$\Sigma^{string} = (S^*; c_1, \dots, c_k, \lambda; g'_1, \dots, g'_{k'}, add, sub_l, sub_r; r'_1, \dots, r'_{k'}, R, =).$$



$$w \in S^*, x \in S, v \in S^* \setminus S \quad \Rightarrow \quad \begin{aligned} add(w, x) &= wx, & add(w, v) &= \lambda, \\ sub_l(\lambda) &= \lambda, & sub_l(wx) &= w, \\ sub_r(\lambda) &= \lambda, & sub_r(wx) &= x, \end{aligned}$$

$$x_1, x_2, \dots \in S \quad \Rightarrow \quad g'_i(x_1, \dots, x_{m_i}) = g_i(x_1, \dots, x_{m_i}), \quad r'_j(x_1, \dots, x_{l_j}) = r_j(x_1, \dots, x_{l_j}),$$

$$(v_1, \dots, v_{m_i}) \in (S^*)^{m_i} \setminus S^{m_i} \quad \Rightarrow \quad g'_i(v_1, \dots, v_{m_i}) = \lambda,$$

$$(v_1, \dots, v_{l_j}) \in (S^*)^{l_j} \setminus S^{l_j} \quad \Rightarrow \quad r'_j(v_1, \dots, v_{l_j}) = false.$$

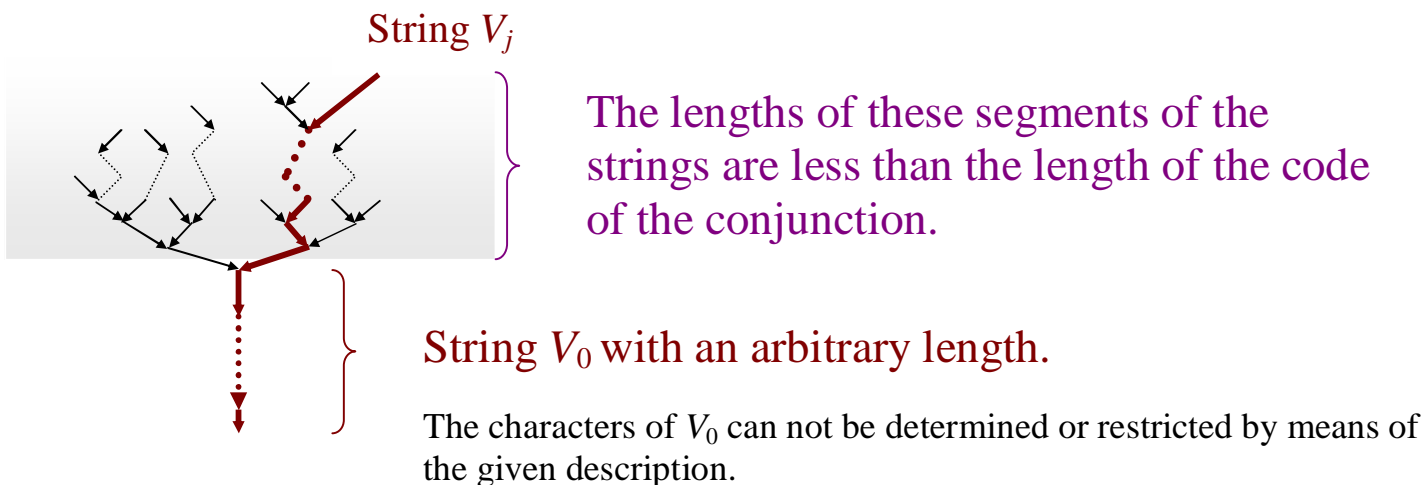
Example: $\Sigma_0 = (\{0, 1\}; 0, 1; ; =) \Rightarrow \Sigma_0^{string} = (\{0, 1\}^*; 0, 1, \lambda; add, sub_l, sub_r; R, =)$

The literals in the formulae in SAT_0^{string} ($= SAT$ for the structure Σ_0^{string}) have the form:

$$\begin{aligned} add(U, V) = W, & \quad R(U), & \quad U = V, \\ add(U, V) \neq W, & \quad \neg R(U), & \quad U \neq V. \end{aligned}$$

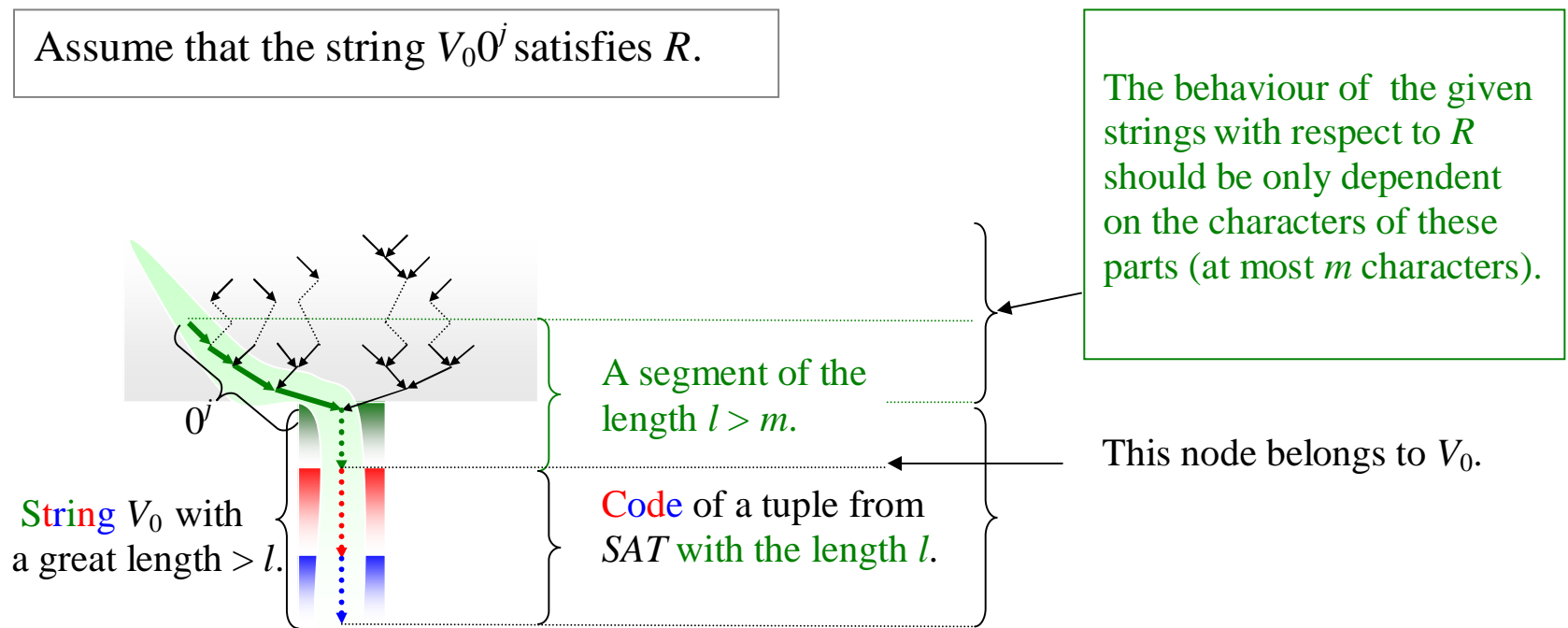
1. A conjunction with *add* and “=” can describe connections between strings like the following.

$$\begin{aligned} & add(V_0, a_0) = V_1 \ \& \ \dots \ \& \ add(V_{j-1}, a_{j-1}) = V_j \quad (a_0, \dots, a_{j-1} \in \{0, 1\}) \\ \Rightarrow & \quad V_j = V_0 a_0 \dots a_{j-1} \ \& \ j < \text{length of the code of the conjunction.} \end{aligned}$$



2. Assume that l is greater than m (= the length of the code of the conjunction) and one of the strings satisfies R .

Assume that the string $V_0 0^j$ satisfies R .



We want to have at most one string w with $R(V_0 w) = true$. Then, we only have to know at most one number $j < m$ with $R(V_0 0^j) = true$. The characters of V_0 do not play a role.

⇒ For each formula:

We restrict the *free* variables to a *domain of short paths* (or strings).

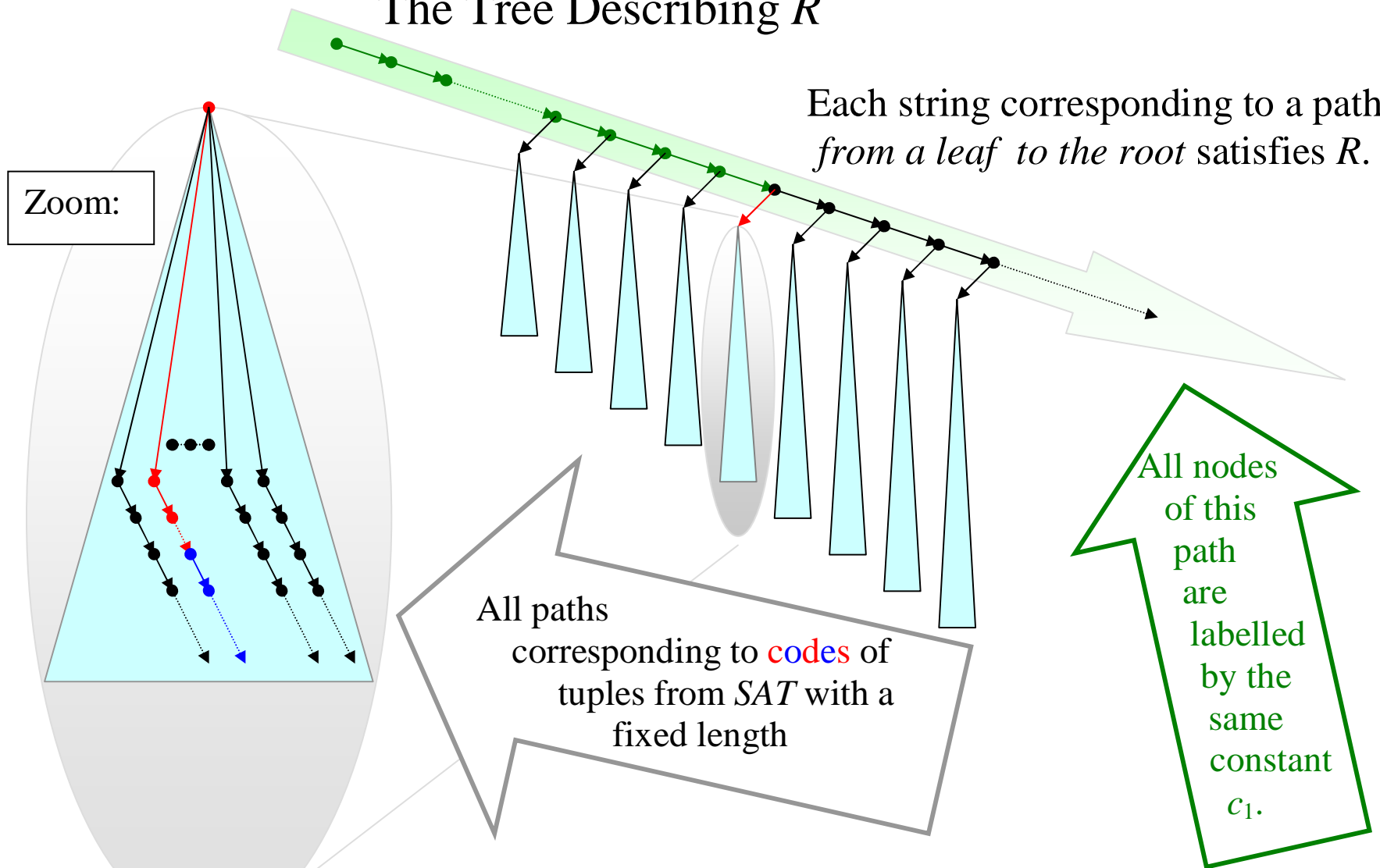
The lengths of the paths in such a domain are only dependent on the length of the code of this formula and of the lengths of the paths in the bounded variables.

⇒ New problem *RES-SAT* (with *RES-SAT* = *SAT*).

⇒ We describe *R* by means of a tree being similar to a tree of computation paths.

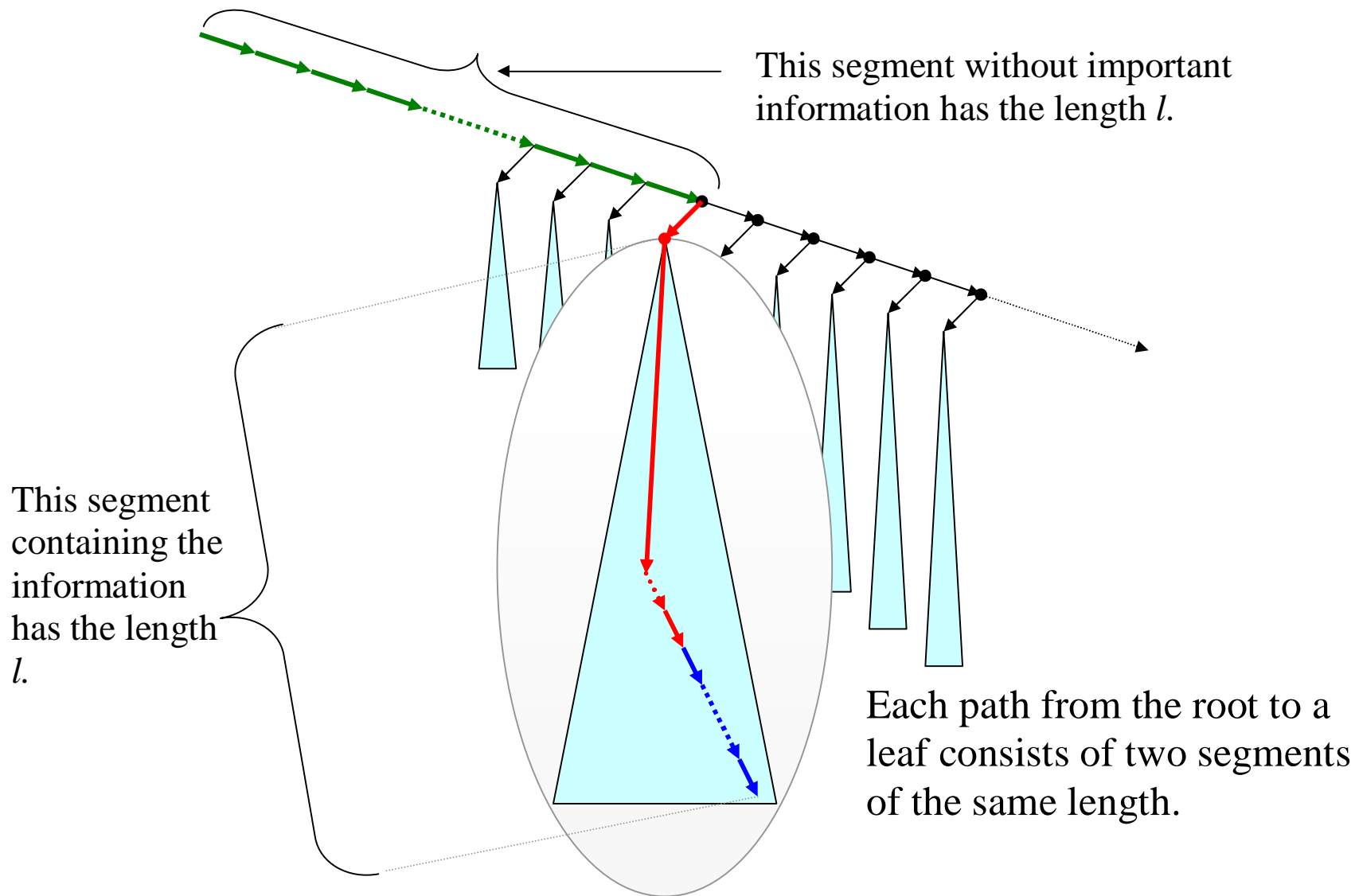
⇒ The behaviour of the paths and their modifications with respect to *R* has to be similar to the behaviour of the inputs traversing computation paths.

The Tree Describing R

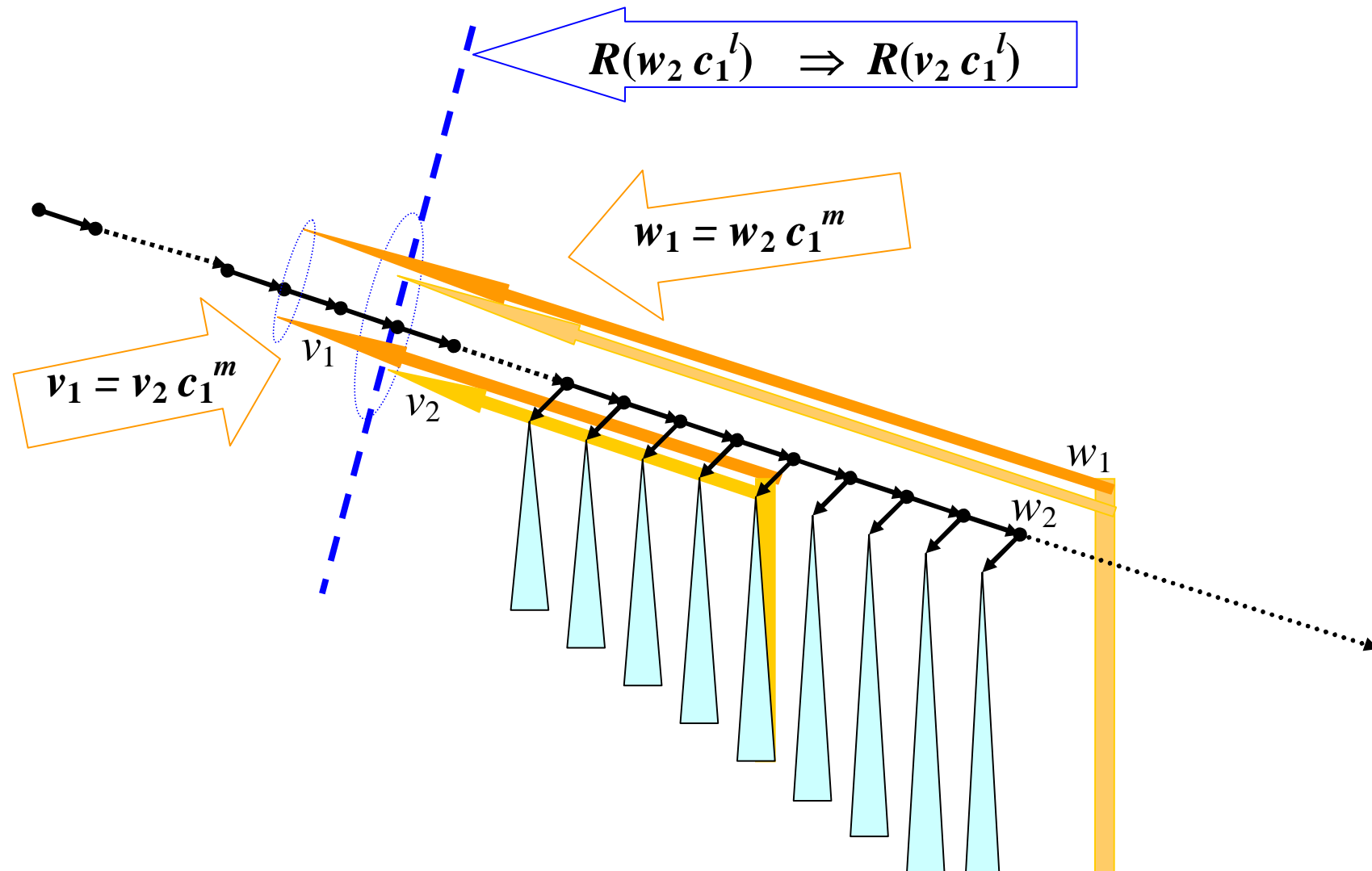


Here we consider an arbitrary structure (\rightarrow infinite alphabet). For Σ_0^{string} , we can take a binary tree.

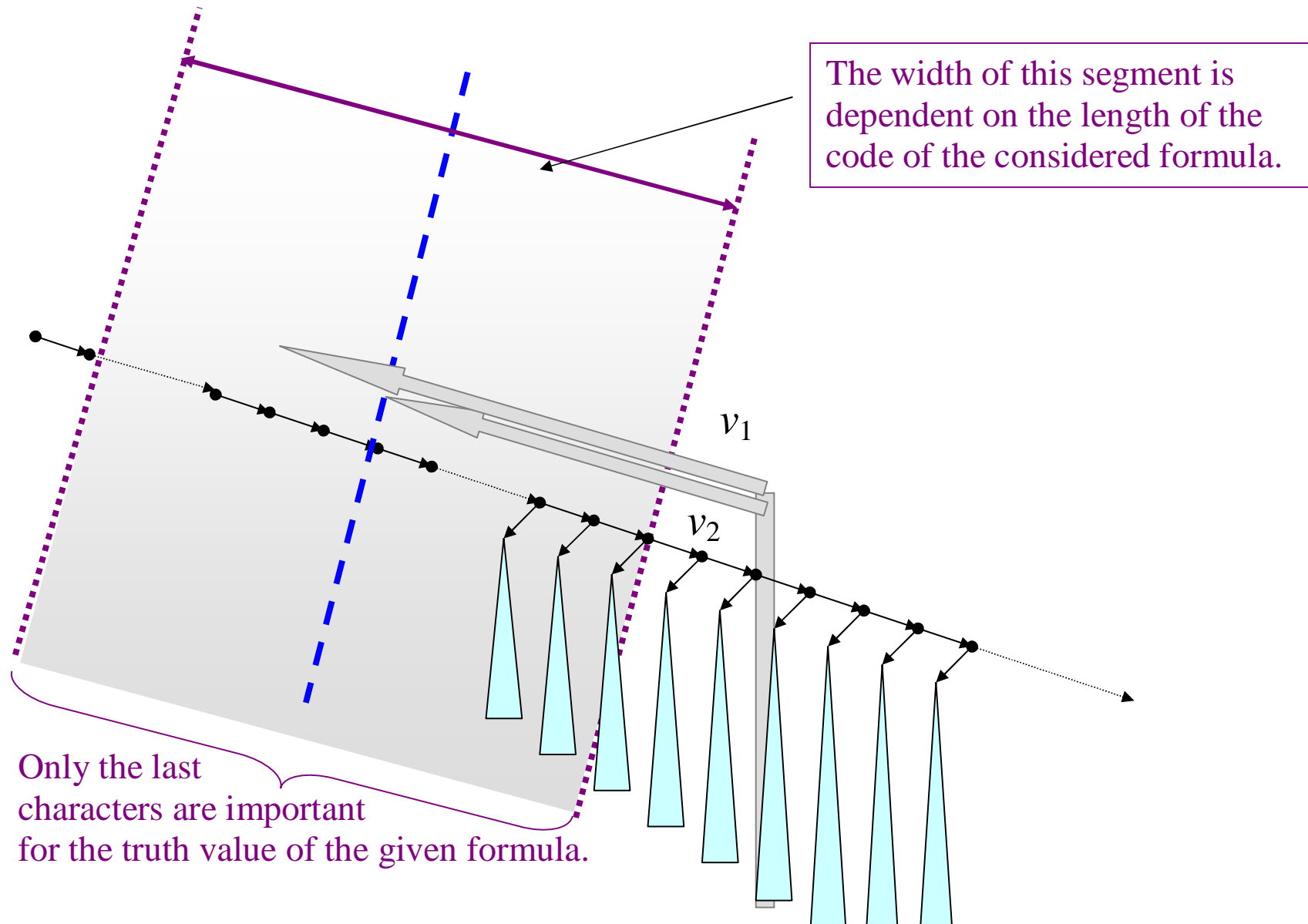
A Path Corresponding to a String Satisfying R



Connections between Different Strings



The Replacement of w_1 by v_1 and of w_2 by v_2 for a Given Formula



Few Definitions for Σ_0^{string}

- $SAT_0^{string} = \{(x_1, \dots, x_n, z_1, \dots, z_m) \mid n, m \in \mathbb{N}^+ \ \& \ (x_1, \dots, x_n) \in (\{0, 1\}^*)^n$
 $\ \& \ (z_1, \dots, z_m) \in \{0, 1\}^m \text{ is the code of a formula } \varphi(X_1, \dots, X_n, Y_1, \dots, Y_l)$
 $\ \& \ \text{there is a } (y_1, \dots, y_l) \text{ with } \Sigma_0^{string} \models \varphi(x_1, \dots, x_n, y_1, \dots, y_l)\}.$

$$M_{j,k} = \{0^i 1^{j-i} s \mid s \in \{0, 1\}^* \ \& \ |s| < k \ \& \ 1 \leq i \leq j - 4\}$$

For all $s \in M = \bigcup_{j,k \in \mathbb{N}} M_{j,k}$, let $R(s) = true$ iff $s = 0^i 1^{j-i} 0^j$ for some $1 \leq i \leq j - 4$.
 $R(\lambda) = false$. For $s \in \{0, 1\}^* \setminus M$ with $|s| > 0$, let $R(s) = true$ iff

$$s = \underbrace{\bar{x}_1 1 1 \bar{x}_2 1 1 \dots \bar{x}_n 1 1 0 z_1 0 z_2 \dots 0 z_m 1 1}_{t \text{ characters}} \underbrace{00 \dots 00}_{t \text{ times "0"}}$$

$t = 2(m_1 + \dots + m_n + m) + 2(n + 1)$

$x_i = x_{i,1} x_{i,2} \dots x_{i,m_i}$
 $\Rightarrow \bar{x}_i = 0 x_{i,1} 0 x_{i,2} \dots 0 x_{i,m_i}$

and

$$\Sigma_0^{string, R'} \models \bigvee \varphi(x_1, \dots, x_n, y_1, \dots, y_l)$$

$$(y_1, \dots, y_l) \in (\{y \mid y \in \{0,1\}^* \ \& \ |y| < 2t\} \cup M_{t,t+l})^l$$

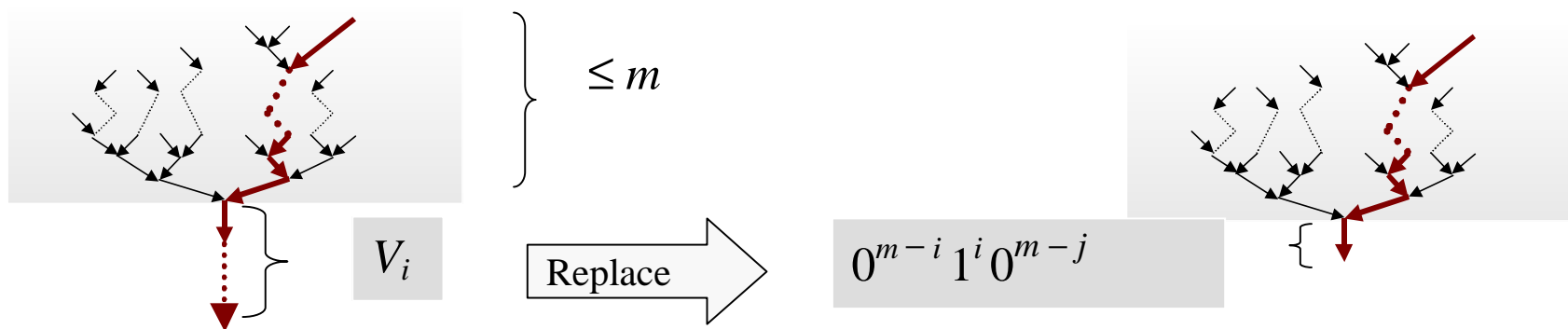
for all R' with $\forall (w \in \{y \mid y \in \{0,1\}^* \ \& \ |y| < 2t\} \cup M_{t,t+l}) (R'(w) = R(w))$ where
 $\Sigma_0^{string, R'} = (\{0, 1\}^*; 0, 1, \lambda; add, sub_l, sub_r; R', =)$ for any relation R' .

Replacements

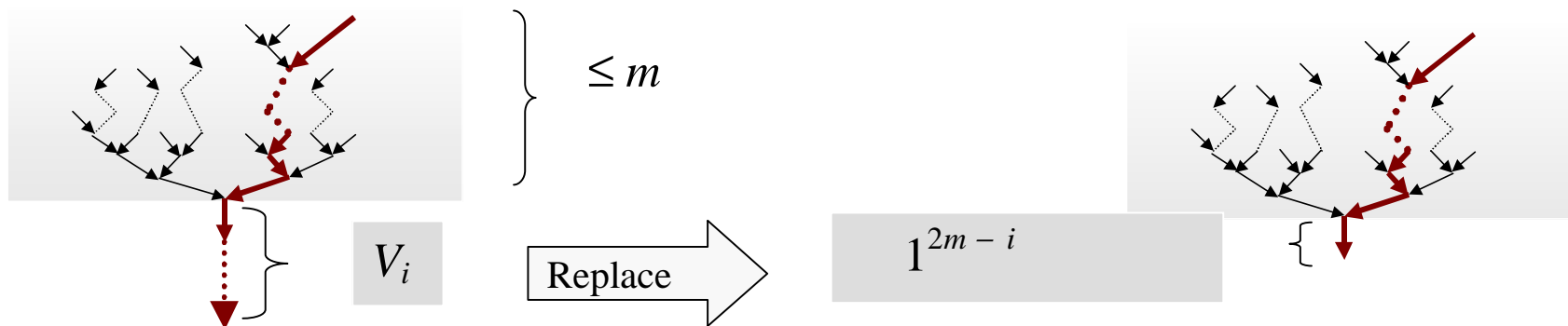
Example – Replacements of long paths for free variables if the considered formula φ does not describe connections between these variables and bounded variables:

1. $|V_i| > m, R(V_i 0^j) = \text{true}$ for some $j < m$.

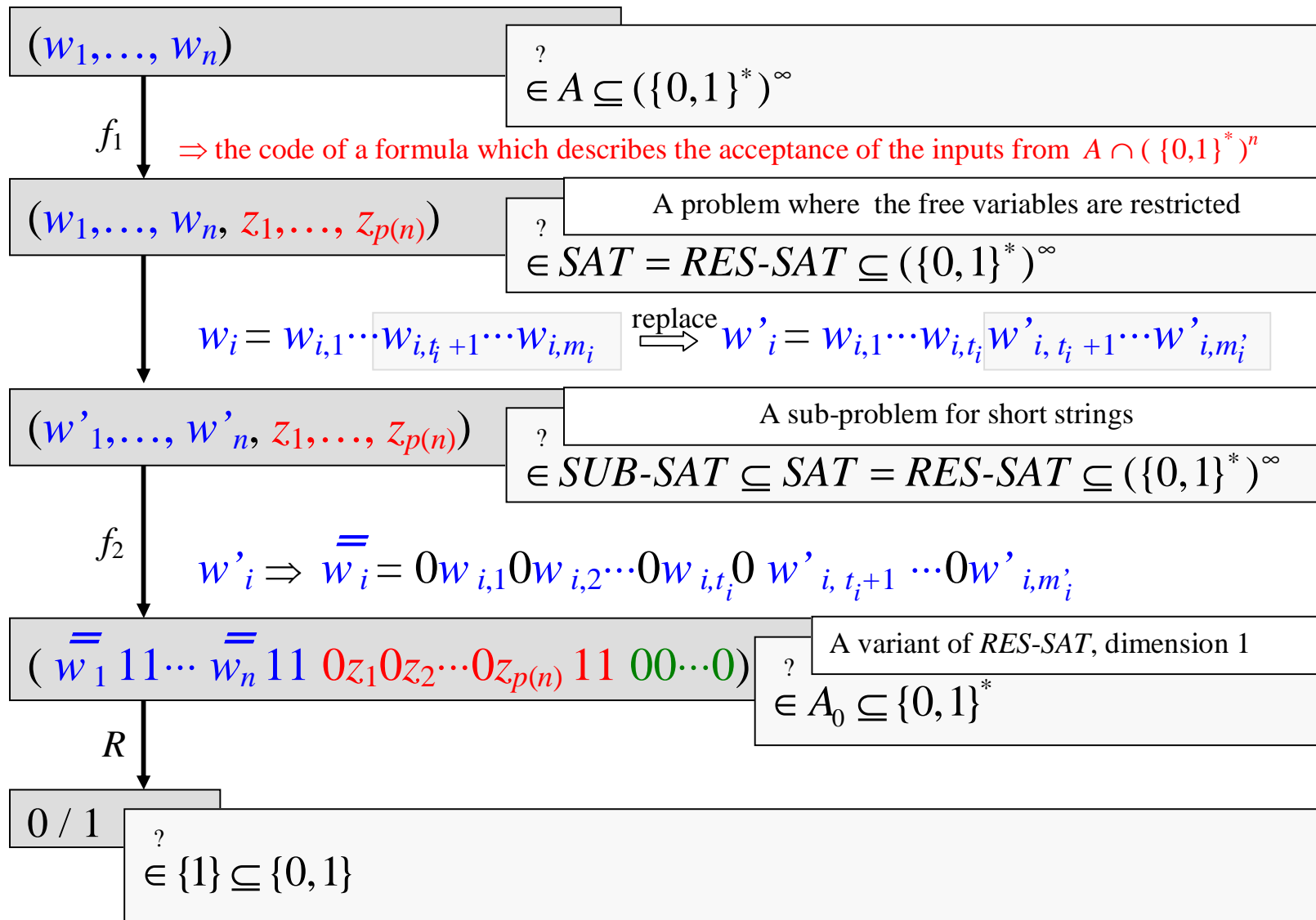
$4 \leq i, m = \text{the length of the code of } \varphi \Rightarrow i < m$



2. $|V_i| > m$ and $R(V_i 0^j) = \text{false}$ for all $j < m$.



The Decision of $A \in NP$ in Polynomial Time



Thank you!

Christine Gaßner

Ernst-Moritz-Arndt-Universität, Greifswald

I thank Mihai Prunescu, Gunther Mainhardt, and Günther Asser for discussions.

I thank Mihai Prunescu and Pascal Koiran for suggestions.

Authors of references: Lenore Blum, Felipe Cucker, Michael Shub, Steve Smale, Armin Hemmerling, Klaus Meer, Pascal Koiran, Bruno Poizat