P = NP for Expansions
Derived from Some Oracles

Christine Gaßner
Greifswald
Our claim:

Any structure can be

► extended to a structure of strings and

► expanded by a relation $R$

to

a structure with $P = NP$. 
P = NP for Expansions Derived from Some Oracles

1. The uniform model of computation

2. Oracles implying $P^A = NP^A$

3. Relations derived from oracles with $P = NP$

4. An ordered structure with $P = NP$

5. Summary
The computation over any structure

Structures (of finite signature):

$$\Sigma = (U; c_1, \ldots, c_u; f_1, \ldots, f_v; R_1, \ldots, R_w, =)$$

- constants
- operations
- relations

Examples:

- $$\mathbb{Z}_2 = \{0, 1\}; 0, 1; +, \cdot ; =)$$ (⇐ Turing machines)
- $$\mathbb{R} = (\mathbb{R}; 0, 1, \ldots; +, -, \cdot ; \leq)$$ (⇐ BSS model)
- $$\Sigma_{\text{string}} = (U^*; \varepsilon, a, b; \text{add}, \text{sub}, \text{sub}_r, =)$$
- $$\Sigma_N = (N; 0, 1; \text{shr}, \text{shl}, \text{inc} \circ \text{shl}; R, \leq)$$
Structures (of finite signature):

\[ \Sigma = (U; c_1, \ldots, c_u; f_1, \ldots, f_v; R_1, \ldots, R_w, =) \]

\[ \text{constants} \quad \text{operations} \quad \text{relations} \]

Examples:

\[ \mathbb{Z}_2 \quad = \quad (\{0, 1\}; 0, 1; +, \cdot; =) \quad (\Rightarrow \text{Turing machines}) \]
\[ \mathbb{R} \quad = \quad (\mathbb{R}; 0, 1, \ldots; +, -, \cdot; \leq) \quad (\Rightarrow \text{BSS model}) \]
\[ \Sigma_{\text{string}} \quad = \quad (U^*; \varepsilon, a, b; \text{add}, \text{sub}_1, \text{sub}_r; =) \]
\[ \Sigma_{\mathbb{N}} \quad = \quad (\mathbb{N}; 0, 1; \text{shr}, \text{shl}, \text{inc} \circ \text{shl}; R, \leq) \]

The uniform model of computation
Oracles implying \( P^A = NP^A \)
Relations derived from oracles with \( P = NP \)
An ordered structure with \( P = NP \)
Summary
The computation instructions

Structure: \( \Sigma = (U; c_1, \ldots, c_w; f_1, \ldots, f_v; R_1, \ldots, R_w, =) \)

Computation:
1: \( Z_k := f_j(Z_{k_1}, \ldots, Z_{k_{m_j}}) \);
2: \( Z_k := c_j \);

Branching:
1: if \( R_j(Z_{k_1}, \ldots, Z_{k_{n_j}}) \) then goto \( l_1 \) else goto \( l_2 \);
2: if \( Z_k = Z_j \) then goto \( l_1 \) else goto \( l_2 \);

Copy:
1: \( Z_{l_k} := Z_{l_j} \);

Index computation:
1: \( l_k := 1 \);
2: \( l_k := l_k + 1 \);
3: if \( l_k = l_j \) then goto \( l_1 \) else goto \( l_2 \);

The uniform model of computation
Oracles implying \( P^A = N^A \)
Relations derived from oracles with \( P = NP \)
An ordered structure with \( P = NP \)

Summary
The computation instructions

Structure: $\Sigma = (U; c_1, \ldots, c_u; f_1, \ldots, f_v; R_1, \ldots, R_w, =)$

Computation:

$l$: $Z_k := f_j(Z_{k1}, \ldots, Z_{km_j});$
$l$: $Z_k := c_j;$

Branching:

$l$: if $R_j(Z_{k1}, \ldots, Z_{km_j})$ then goto $l_1$ else goto $l_2;$
$l$: if $Z_k = Z_j$ then goto $l_1$ else goto $l_2;$

Copy:

$l$: $Z_{ik} := Z_j;$

Index computation:

$I_k := 1; I_k := I_k + 1; \text{ if } I_k = I_j \text{ then goto } l_1 \text{ else goto } l_2;$

The uniform model of computation
Oracles implying $P^A = N^P^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$

Summary
The uniform model of computation
Oracles implying $P^A = NP^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$
Summary

Structure: $\Sigma = (U; c_1, \ldots, c_w; f_1, \ldots, f_v; R_1, \ldots, R_w, =)$

Computation:
- $l$: $Z_k := f_j(Z_{k1}, \ldots, Z_{km_j})$
- $l$: $Z_k := c_j$

Branching:
- $l$: if $R_j(Z_{k1}, \ldots, Z_{km_j})$ then goto $l_1$ else goto $l_2$
- $l$: if $Z_k = Z_j$ then goto $l_1$ else goto $l_2$

Copy:
- $l$: $Z_{l_k} := Z_{l_j}$

Index computation:
- $I_{l_k} := 1$
- $I_{l_k} := I_{l_k} + 1$
- if $I_{l_k} = I_j$ then goto $l_1$ else goto $l_2$
The computation instructions

Structure: $\Sigma = (U; c_1, \ldots, c_v; f_1, \ldots, f_v; R_1, \ldots, R_w, =)$

Computation: 
\begin{align*}
  l: & \quad Z_k := f_j(Z_{k1}, \ldots, Z_{kmj}); \\
  l: & \quad Z_k := c_j;
\end{align*}

Branching: 
\begin{align*}
  l: & \quad \text{if } R_j(Z_{k1}, \ldots, Z_{kmj}) \text{ then goto } l_1 \text{ else goto } l_2; \\
  l: & \quad \text{if } Z_k = Z_j \text{ then goto } l_1 \text{ else goto } l_2;
\end{align*}

Copy: 
\begin{align*}
  l: & \quad Z_{jk} := Z_{lj};
\end{align*}

Index computation: 
\begin{align*}
  I_k & := 1; \quad I_k := I_k + 1; \quad \text{if } I_k = I_j \text{ then goto } l_1 \text{ else goto } l_2;
\end{align*}
The computation instructions

Structure: \( \Sigma = (U; c_1, \ldots, c_u; f_1, \ldots, f_v; R_1, \ldots, R_w; =) \)

Computation:
\[
\begin{align*}
l & : \ Z_k := f_j(Z_{k1}, \ldots, Z_{kmj}); \\
l & : \ Z_k := c_j;
\end{align*}
\]

Branching:
\[
\begin{align*}
l & : \ \text{if } R_j(Z_{k1}, \ldots, Z_{kmj}) \ \text{then goto } l_1 \ \text{else goto } l_2; \\
l & : \ \text{if } Z_k = Z_j \ \text{then goto } l_1 \ \text{else goto } l_2;
\end{align*}
\]

Copy:
\[
l : \ Z_{l_k} := Z_{l_j};
\]

Index computation:
\[
I_k := 1; \quad I_k := I_k + 1; \quad \text{if } I_k = I_j \ \text{then goto } l_1 \ \text{else goto } l_2;
\]

The uniform model of computation
Oracles implying \( P^A = \text{NP}^A \)
Relations derived from oracles with \( P = \text{NP} \)
An ordered structure with \( P = \text{NP} \)

Summary
### Deterministic and non-deterministic machines

<table>
<thead>
<tr>
<th>The input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>((Z_1, \ldots, Z_n) := (x_1, \ldots, x_n) \in \bigcup_{i \geq 1} U^i)</td>
</tr>
</tbody>
</table>

Every \(u \in U\) can be stored in one register.

<table>
<thead>
<tr>
<th>The input and guessing:</th>
</tr>
</thead>
<tbody>
<tr>
<td>((Z_1, \ldots, Z_n, Z_{n+1}, \ldots, Z_{n+m}) := (x_1, \ldots, x_n, y_1, \ldots, y_m) \in \bigcup_{i \geq 1} U^i)</td>
</tr>
</tbody>
</table>

Arbitrary elements can be guessed!
The input:

\[(Z_1, \ldots, Z_n) := (x_1, \ldots, x_n) \in \bigcup_{i \geq 1} U^i\]

Every \(u \in U\) can be stored in one register.

The input and guessing:

\[(Z_1, \ldots, Z_n, Z_{n+1}, \ldots, Z_{n+m}) := (x_1, \ldots, x_n, y_1, \ldots, y_m) \in \bigcup_{i \geq 1} U^i\]

Arbitrary elements can be guessed!
Deterministic and non-deterministic machines

The input:

\[(Z_1, \ldots, Z_n) := (x_1, \ldots, x_n) \in \bigcup_{i \geq 1} U^i\]

Every \(u \in U\) can be stored in one register.

The input and guessing:

\[(Z_1, \ldots, Z_n, Z_{n+1}, \ldots, Z_{n+m}) := (x_1, \ldots, x_n, y_1, \ldots, y_m) \in \bigcup_{i \geq 1} U^i\]

⇒ Arbitrary elements can be guessed!
The input:

\[(Z_1, \ldots, Z_n) := (x_1, \ldots, x_n) \in \bigcup_{i \geq 1} U^i\]

Every \(u \in U\) can be stored in one register.

The input and guessing:

\[(Z_1, \ldots, Z_n, Z_{n+1}, \ldots, Z_{n+m}) := (x_1, \ldots, x_n, y_1, \ldots, y_m) \in \bigcup_{i \geq 1} U^i\]

Arbitrary elements can be guessed!
The computation in polynomial time for the uniform model

Computation in polynomial time:

For any machine $M$ there is some polynomial $p_M$ such that for all $(x_1, \ldots, x_n)$

\[ M \text{ halts for } x = (x_1, \ldots, x_n) \text{ within } p_M(n) \text{ steps.} \]

The execution of one operation $= \text{ one time unit.}$

\[ P_\Sigma \subseteq NP_\Sigma \quad P_\Sigma \subseteq \text{DEC}_\Sigma \rightarrow NP_\Sigma \nsubseteq \text{DEC}_\Sigma \Rightarrow P_\Sigma \neq NP_\Sigma \]
Oracle machines

Structure: $\Sigma = (U; a, b, c_3, \ldots, c_u; f_1, \ldots, f_r; R_1, \ldots, R_w, =)$

Oracle: $A \subseteq U^\infty = \text{df } \bigcup_{i \geq 1} U^i$

Oracle query: $l$: if $(Z_1, \ldots, Z_{I_1}) \in A$ then goto $l_1$ else goto $l_2$;

The length can be computed by $I_1 := 1; \quad I_1 := I_1 + 1; \quad \ldots$

Proposition (Baker, Gill, and Solovay; Emerson; ...):

For any structure $\Sigma$, there is some oracle $O_\Sigma$ such that $P^O_\Sigma = NP^O_\Sigma$.
Oracle machines

Structure: $\Sigma = (U; a, b, c_3, \ldots, c_u; f_1, \ldots, f_v; R_1, \ldots, R_w, =)$

Oracle: $A \subseteq U^\infty = \bigcup_{i \geq 1} U^i$

Oracle query: $l: \text{if } (Z_1, \ldots, Z_{I_1}) \in A \text{ then goto } l_1 \text{ else goto } l_2$

The length can be computed by $I_1 := 1; \quad I_1 := I_1 + 1; \ldots$

Proposition (Baker, Gill, and Solovay; Emerson; ...):

For any structure $\Sigma$, there is some oracle $O_\Sigma$ such that $P^\Sigma_{O_\Sigma} = NP^\Sigma_{O_\Sigma}$.
The oracle $O_{\Sigma}$ with $P_{\Sigma}^{O_{\Sigma}} = NP_{\Sigma}^{O_{\Sigma}}$

A universal oracle:

$$O_{\Sigma} = \{ (x, \text{Code}(M), b, \ldots, b) \mid x \in U^\infty \text{ and } M \text{ is a non-deterministic } O_{\Sigma} \text{-machine}$$

$$\& \text{ } M(x) \downarrow^t \}\}$$

$M$ accepts $x = (x_1, \ldots, x_n) \in U^\infty$ within $t$ steps.

The uniform model of computation
Oracles implying $P^A = NP^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$
Summary
Why do we use strings?

Our goal:
- A relation $R$ allows to decide whether $z \in O_\Sigma$.
- $R$ can be defined recursively.

Problems:
- Each relation has a fixed arity.
- $O_\Sigma$ contains tuples of any length.
- For many structures: The tuples of arbitrary length cannot be encoded by tuples of fixed length.

A solution:
- Strings.
Why do we use strings?

Our goal:
- A relation $R$ allows to decide whether $z \in O_\Sigma$.
- $R$ can be defined recursively.

Problems:
- Each relation has a fixed arity.
- $O_\Sigma$ contains tuples of any length.
- For many structures:
  The tuples of arbitrary length cannot be encoded by tuples of fixed length.

A solution:
- Strings.

The uniform model of computation
Oracles implying $P^A = NP^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$
Summary
Our goal:
- A relation $R$ allows to decide whether $z \in O_\Sigma$.
- $R$ can be defined recursively.

Problems:
- Each relation has a fixed arity.
- $O_\Sigma$ contains tuples of any length.
- For many structures:
The tuples of arbitrary length cannot be encoded by tuples of fixed length.

A solution:
- Strings.
Our goal:
► A relation \( R \) allows to decide whether \( z \in O_\Sigma \).
► \( R \) can be defined recursively.

Problems:
► Each relation has a fixed arity.
► \( O_\Sigma \) contains tuples of any length.
► For many structures:
The tuples of arbitrary length cannot be encoded by tuples of fixed length.

A solution:
► Strings.
Structures over strings

\[ \Sigma = (U^*; \varepsilon, a, b, c_3, \ldots, c_u; \text{add}, \text{sub}_1, \text{sub}_r, f_1, \ldots, f_v; R_1, \ldots, R_w, R, =) \]

\[ s = d_1 \cdots d_k \in U^* \]
\[ (d_1, \ldots, d_k) \in U^k \subset U^\infty \]

- stored in one register
- stored in \( k \) registers

\[ \text{add}(s, d) = sd \]
\[ \text{sub}_1(sd) = s \]
\[ \text{sub}_r(sd) = d \]

\[ s \in U^*, d \in U \]

\[ R_i \subseteq U^n; \]
\[ f_i(s_{1}, \ldots, s_{m}) = \varepsilon \quad \text{if} \quad |s_j| > 1 \quad \text{for some} \ j \]

The uniform model of computation
Oracles implying \( P^A = NP^A \)
Relations derived from oracles with \( P = NP \)
An ordered structure with \( P = NP \)
Summary
Structures over strings

\[ \Sigma = (U^*; \varepsilon, a, b, c_3, \ldots, c_u; \text{add}, \text{sub}_1, \text{sub}_r, f_1, \ldots, f_v; R_1, \ldots, R_w, R, =) \]

\[ s = d_1 \cdots d_k \in U^* \]

\[ (d_1, \ldots, d_k) \in U^k \subset U^{\infty} \]

stored in one register

stored in \( k \) registers

\[ \text{add}(s, d) = sd \quad \text{sub}_1(sd) = s \quad \text{sub}_r(sd) = d \]

\[ s \in U^*, d \in U \]

\[ R_i \subseteq U^n; \]

\[ f_i(s_1, \ldots, s_m) = \varepsilon \quad \text{if } |s_j| > 1 \text{ for some } j \]
The uniform model of computation
Oracles implying $P^A = NP^A$

Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$

Summary

Structures over strings

$\Sigma = (U^*; \varepsilon, a, b, c_3, \ldots, c_u; \text{add}, \text{sub}_1, \text{sub}_\Gamma, f_1, \ldots, f_v; \ R_1, \ldots, R_w, R, =)$

$s = d_1 \ldots d_k \in U^*$
$(d_1, \ldots, d_k) \in U^k \subset U^\infty$

stored in one register
stored in $k$ registers

$\text{add}(s, d) = sd$
$\text{sub}_1(sd) = s$
$\text{sub}_\Gamma(sd) = d$

$s \in U^*; d \in U$

$R_i \subseteq U^{n_i}$

$f_i(s_{i_1}, \ldots, s_{m_i}) = \varepsilon$ if $|s_j| > 1$ for some $j$
Example: \( \Sigma = (\{a, b\}^* ; \varepsilon, a, b; \text{add}, \text{sub}); =) \)

**Definition:** \( s \subseteq r \iff \text{sub}_1(r) = s. \)

**Lemma.** For \( k \) steps of a machine holds:

1. The input values, the guesses, and the new computed values form maximal chains \( s_1 \subseteq s_2 \cdots \subseteq s_k \).
2. The maximal chains form trees. Every tree has only one minimal element.
3. The predecessors \( r \subseteq s \) of the minimal elements \( s \) are not computed.

**Corollary:**

The minimal elements can be replaced without changing the computation path.

---

Christine Gaßner  
gassnerc@uni-greifswald.de
Computation over strings

Example: \( \Sigma = (\{a, b\}^*; \varepsilon, a, b; \text{add}, \text{sub}; =) \)

Definition: \( s \subseteq_r p \iff \text{sub}_1(p) = s \).

Lemma. For \( k \) steps of a machine holds:
1. The input values, the guesses, and the new computed values form maximal chains \( x_1 \subseteq_1 \cdots \subseteq_1 x_k \).
2. The maximal chains form trees. Every tree has only one minimal element.
3. The predecessors \( r \subseteq_1 x_1 \) of the minimal elements \( x_1 \) are not computed.

Corollary:
The minimal elements can be replaced without changing the computation path.
Computation over strings

Example: $\Sigma = (\{a, b\}^*; \varepsilon, a, b; \text{add, sub}_1; =)$

1. $r = sa \quad \text{add}(s, a) = r \quad \text{sub}_1(r) = s$

Definition: $s \subset_{r_1} r \iff \text{sub}_1(r) = s$.

Lemma. For $k$ steps of a machine holds:

1. The input values, the guesses, and the new computed values form maximal chains $s_1 \subset \cdots \subset s_k$.
2. The maximal chains form trees. Every tree has only one minimal element.
3. The predecessors $r \subset s_1$ of the minimal elements $s_1$ are not computed.

Corollary:
The minimal elements can be replaced without changing the computation path.
**Computation over strings**

**Example:** $\Sigma = (\{a, b\}^* ; \varepsilon, a, b; add, sub, =)$

![Diagram showing computation over strings]

1. $r = sa \quad add(s, a) = r \quad sub_1(r) = s$

2. $r = sb \quad add(s, b) = r \quad sub_1(r) = s$

**Definition:** $s \subset r \iff sub_1(r) = s$.

**Lemma.** For $k$ steps of a machine holds:

1. The input values, the guesses, and the new computed values form maximal chains $x_1 \subset \ldots \subset x_k$.
2. The maximal chains form trees. Every tree has only one minimal element.
3. The predecessors $r \subset x_1$ of the minimal elements $x_1$ are not computed.

**Corollary:**

The minimal elements can be replaced without changing the computation path.

---

The uniform model of computation
Oracles implying $P^A = NP^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$

**Summary**

Christine Gaßner  

gassnerc@uni-greifswald.de
Computation over strings

Example: \( \Sigma = (\{a, b\}^*; \varepsilon, a, b; add, sub); =) \)

\[
1) \quad r = sa \quad add(s, a) = r \quad sub_1(r) = s
\]

\[
2) \quad r = sb \quad add(s, b) = r \quad sub_1(r) = s
\]

Definition: \( s \subseteq_1 r \iff sub_1(r) = s. \)

Lemma. For \( t \) steps of a machine holds:

1. The input values, the guesses, and the new computed values form maximal chains \( s_1 \subseteq \cdots \subseteq s_t. \)
2. The maximal chains form trees. Every tree has only one minimal element.
3. The predecessors \( r \subseteq s_1 \) of the minimal elements \( s_1 \) are not computed.

Corollary: The minimal elements can be replaced without changing the computation path.
Computation over strings

Example: \( \Sigma = (\{a, b\}^*; \varepsilon, a, b; add, sub; =) \)

Definition: \( s \subseteq_1 r \iff sub_1(r) = s. \)

Lemma. For \( t \) steps of a machine holds:

1. The input values, the guesses, and the new computed values form maximal chains \( s_1 \subseteq_1 \ldots \subseteq_1 s_t. \)
2. The maximal chains form trees. Every tree has only one minimal element.
3. The predecessors \( r \subseteq_1 s_1 \) of the minimal elements \( s_1 \) are not computed.

Corollary:
The minimal elements can be replaced without changing the computation path.
Computation over strings

Example: $\Sigma = (\{a, b\}^* ; \varepsilon, a, b; \text{add}, \text{sub}, ; =)$

Definition: $s \subseteq_1 r \iff \text{sub}_1(r) = s.$

Lemma. For $t$ steps of a machine holds:
1. The input values, the guesses, and the new computed values form maximal chains $s_1 \subseteq_1 \cdots \subseteq_1 s_k$.
2. The maximal chains form trees. Every tree has only one minimal element.
3. The predecessors $r \subseteq_1 s_1$ of the minimal elements $s_1$ are not computed.

Corollary:
The minimal elements can be replaced without changing the computation path.

The uniform model of computation
Oracles implying $P^A = NP^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$

Summary
Computation over strings

Example: \( \Sigma = (\{a, b\}^*; \varepsilon, a, b; \text{add}, \text{sub}; =) \)

Definition: \( s \subset_{1} r \iff \text{sub}_1(r) = s \).

Lemma. For \( t \) steps of a machine holds:
1. The input values, the guesses, and the new computed values form maximal chains \( s_1 \subset_{1} \cdots \subset_{1} s_k \).
2. The maximal chains form trees. Every tree has only one minimal element.
3. The predecessors \( r \subset_{1} s_1 \) of the minimal elements \( s_1 \) are not computed.

Corollary:
The minimal elements can be replaced without changing the computation path.

The uniform model of computation
Oracles implying \( P^A = \text{NP}^A \)
Relations derived from oracles with \( P = \text{NP} \)
An ordered structure with \( P = \text{NP} \)

Summary
Computation over strings

Example: $\Sigma = (\{a, b\}^*; \varepsilon, a, b; add, sub; =)$

Definition: $s \subseteq_1 r \iff \text{sub}_i(r) = s$.

Lemma. For $t$ steps of a machine holds:
1. The input values, the guesses, and the new computed values form maximal chains $s_1 \subseteq_1 \cdots \subseteq_1 s_k$.
2. The maximal chains form trees. Every tree has only one minimal element.
3. The predecessors $r \subseteq_1 s_1$ of the minimal elements $s_1$ are not computed.

Corollary: The minimal elements can be replaced without changing the computation path.

Christine Gaßner  gassnerc@uni-greifswald.de
Computation over strings

Example: \( \Sigma = (\{a, b\}^*; \varepsilon, a, b; add, sub; \}) \)

Definition: \( s \subseteq_1 r \iff \text{sub}_1(r) = s \).

Lemma. For \( t \) steps of a machine holds:
1. The input values, the guesses, and the new computed values form maximal chains \( s_1 \subseteq_1 \cdots \subseteq_1 s_k \).
2. The maximal chains form trees. Every tree has only one minimal element.
3. The predecessors \( r \subseteq_1 s_1 \) of the minimal elements \( s_1 \) are not computed.

Corollary:
The minimal elements can be replaced without changing the computation path.

The uniform model of computation
Oracles implying \( P^A = NP^A \)
Relations derived from oracles with \( P = NP \)
An ordered structure with \( P = NP \)
Summary
Computation over strings

Example: $\Sigma = (\{a, b\}^* ; \varepsilon, a, b; \text{add}, \text{sub}, ; =)$

Definition: $s \subseteq_r r \iff \text{sub}_1(r) = s$.

Lemma. For $i$ steps of a machine holds:
1. The input values, the guesses, and the new computed values form maximal chains $s_1 \subseteq_1 \cdots \subseteq_1 s_k$.
2. The maximal chains form trees. Every tree has only one minimal element.
3. The predecessors $r \subseteq s_1$ of the minimal elements $s_1$ are not computed.

Corollary: The minimal elements can be replaced without changing the computation path.

The uniform model of computation
Oracles implying $P^A = NP^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$
Summary
Computation over strings

Example: $\Sigma = (\{a, b\}^*; \varepsilon, a, b; \text{add}, \text{sub}; =)$

Definition: $s \subseteq_1 r \iff \text{sub}_1(r) = s$.

Lemma. For $i$ steps of a machine holds:
1. The input values, the guesses, and the new computed values form maximal chains $s_1 \subseteq_1 \cdots \subseteq_1 s_k$.
2. The maximal chains form trees. Every tree has only one minimal element.
3. The predecessors $r \subseteq_1 s_1$ of the minimal elements $s_1$ are not computed.

Corollary: The minimal elements can be replaced without changing the computation path.

The uniform model of computation
Oracles implying $P^A = NP^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$
Summary
Computation over strings

Example: \( \Sigma = (\{a, b\}^*; \varepsilon, a, b; \text{add}, \text{sub}, \text{=} ) \)

Definition: \( s_1 \subseteq r \iff \text{sub}_1(r) = s \).

Lemma. For \( t \) steps of a machine holds:

1. The input values, the guesses, and the new computed values form maximal chains \( s_1 \subseteq s_2 \cdots \subseteq s_k \).
2. The maximal chains form trees. Every tree has only one minimal element.
3. The predecessors \( r \subseteq s_1 \) of the minimal elements \( s_1 \) are not computed.

Corollary:

The minimal elements can be replaced without changing the computation path.

Christine Gaßner  gassnerc@uni-greifswald.de
Computation over strings

Example: $\Sigma = (\{a, b\}^*; \varepsilon, a, b; \text{add}, \text{sub}; =)$

Definition: $s \subset_1 r \iff \text{sub}_1(r) = s$.

Lemma. For $t$ steps of a machine holds:

1. The input values, the guesses, and the new computed values form maximal chains $s_1 \subset \cdots \subset s_k$.
2. The maximal chains form trees. Every tree has only one minimal element.
3. The predecessors $r \subset_1 s_1$ of the minimal elements $s_1$ are not computed.

Corollary:

The minimal elements can be replaced without changing the computation path.

The uniform model of computation
Oracles implying $P^A = NP^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$
Summary
Why do we pad the codes?

Our goal:
- Structures $\Sigma$ with $NP_\Sigma \subseteq DEC_\Sigma$.

Problems:
- Arbitrary strings can be guessed.
- A new $R$ could imply $H_{\Sigma_R} \subseteq NP_{\Sigma_R} \setminus DEC_{\Sigma_R}$ for the halting problem $H_{\Sigma_R}$.

Solution:
- Padding strings: $R(\Sigma) \Rightarrow (\exists r \in U^*)(\Sigma = ra^{|r|})$.

It allows to replace:
- long inputs and guesses
- by short strings over $\{a, b\}$.
Why do we pad the codes?

Our goal:
► Structures \( \Sigma \) with \( NP_\Sigma \subseteq DEC_\Sigma \).

Problems:
► Arbitrary strings can be guessed.
► A new \( R \) could imply \( H_{\Sigma_R} \subseteq NP_{\Sigma_R} \setminus DEC_{\Sigma_R} \) for the halting problem \( H_{\Sigma_R} \).

Solution:
► Padding strings:
\[
R(s) \Rightarrow (\exists r \in U^*) (s = ra^{|s|}).
\]

It allows to replace
- long inputs and guesses
- by short strings over \( \{a, b\} \).

The uniform model of computation
Oracles implying \( P^A = NP^A \)
Relations derived from oracles with \( P = NP \)
An ordered structure with \( P = NP \)
Summary
Why do we pad the codes?

Our goal:
- Structures $\Sigma$ with $\text{NP}_\Sigma \subseteq \text{DEC}_\Sigma$.

Problems:
- Arbitrary strings can be guessed.
- A new $R$ could imply $H_{\Sigma_R} \in \text{NP}_\Sigma \setminus \text{DEC}_\Sigma$ for the halting problem $H_{\Sigma_R}$.

Solution:
- Padding strings:
  
  $R(s) \Rightarrow (\exists r \in U^*) (s = ra^{\left| r \right|})$.

It allows to replace
- long inputs and guesses
- by short strings over $\{a, b\}$.
Why do we pad the codes?

Our goal:
- Structures $\Sigma$ with $\text{NP}_\Sigma \subseteq \text{DEC}_\Sigma$.

Problems:
- Arbitrary strings can be guessed.
- A new $R$ could imply $\text{H}_{\Sigma_R} \in \text{NP}_{\Sigma_R} \setminus \text{DEC}_{\Sigma_R}$ for the halting problem $\text{H}_{\Sigma_R}$.

Solution:
- Padding strings:
  \[ R(s) \Rightarrow (\exists r \in U^*)(s = ra^{\mid r\mid}). \]

It allows to replace
- long inputs and guesses
- by short strings over $\{a, b\}$.

The paths corresponding to the strings satisfying $R$

The uniform model of computation
- Oracles implying $P^A = N^A$
- Relations derived from oracles with $P = NP$
- An ordered structure with $P = NP$

Summary

Christine Gaßner    gassnerc@uni-greifswald.de
Why do we pad the codes?

Our goal:
- Structures $\Sigma$ with $\text{NP}_\Sigma \subseteq \text{DEC}_\Sigma$.

Problems:
- Arbitrary strings can be guessed.
- A new $R$ could imply $H_{\Sigma_R} \in \text{NP}_{\Sigma_R} \setminus \text{DEC}_{\Sigma_R}$ for the halting problem $H_{\Sigma_R}$.

Solution:
- Padding strings:
  $$R(s) \Rightarrow (\exists r \in U^*)(s = ra^{|r|}).$$

It allows to replace
- long inputs and guesses
- by short strings over $\{a, b\}$. 

The uniform model of computation
Oracles implying $P^A = NP^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$

Summary
Why do we pad the codes?

Our goal:
- Structures $\Sigma$ with $\text{NP}_\Sigma \subseteq \text{DEC}_\Sigma$.

Problems:
- Arbitrary strings can be guessed.
- A new $R$ could imply $H_{\Sigma R} \in \text{NP}_{\Sigma R} \setminus \text{DEC}_{\Sigma R}$ for the halting problem $H_{\Sigma R}$.

Solution:
- Padding strings:
  $$R(s) \Rightarrow (\exists r \in U^*) (s = ra| r |).$$

  It allows to replace
  - long inputs and guesses
  - by short strings over $\{a, b\}$.
The new relation $R$

$\Sigma = (U^*; a, b, c_3, \ldots, c_n; f_1, \ldots, f_v; R_1, \ldots, R_w, =)$

$\Sigma_R = \text{Expansion of } \Sigma \text{ by } R$

A universal oracle:

Let $W_\Sigma \subset U^\infty$ with $P_\Sigma^{W_\Sigma} = NP_\Sigma^{W_\Sigma}$ (derived from $O_\Sigma$).

The relation $R$:

$r_1 \ldots r_k a | r_1 \ldots r_k | \in R \iff (r_1, \ldots, r_k) \in W_\Sigma$

Theorem:

$P_{\Sigma_R} = NP_{\Sigma_R}$. 

The uniform model of computation

Oracles implying $P^A = NP^A$

Relations derived from oracles with $P = NP$

An ordered structure with $P = NP$

Summary
The new relation $R$

The uniform model of computation
Oracles implying $P^A = NP^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$

Summary

$\Sigma = (U^*; a, b, c_3, \ldots, c_u; f_1, \ldots, f_v; R_1, \ldots, R_w, =)$

$\Sigma_R = \text{Expansion of } \Sigma \text{ by } R$

A universal oracle:

Let $W_\Sigma \subset U^\infty$ with $P^\Sigma_{W_\Sigma} = NP^\Sigma_{W_\Sigma}$ (derived from $O_\Sigma$).

The relation $R$:

$r_1 \cdots r_k a | r_1 \cdots r_k | \in R \iff (r_1, \ldots, r_k) \in W_\Sigma$

Theorem:

$P^\Sigma_{\Sigma_R} = NP^\Sigma_{\Sigma_R}$.
The new relation $R$

The uniform model of computation
Oracles implying $P^A = NP^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$

Summary

$\Sigma = (U^*; a, b, c_3, \ldots, c_n; f_1, \ldots, f_v; R_1, \ldots, R_w, =)$
$\Sigma^R = \text{Expansion of } \Sigma \text{ by } R$

A universal oracle:

Let $W_{\Sigma} \subset U^\infty$ with $P_{\Sigma}^{W_{\Sigma}} = NP_{\Sigma}^{W_{\Sigma}}$ (derived from $O_{\Sigma}$).

The relation $R$:

$$r_1 \cdots r_k a |r_1 \cdots r_k| \in R \iff (r_1, \ldots, r_k) \in W_{\Sigma}$$

Theorem:

$P_{\Sigma^R} = NP_{\Sigma^R}$. 

Christine Gaßner  gassnerc@uni-greifswald.de
Proof of $P_{\Sigma^R} = NP_{\Sigma^R}$ by a reduction.

\[ \text{UNI} = \{(x_1, \ldots, x_n, \text{Code}(M), b, \ldots, b) \mid x \in (U^*)^\infty \land M \text{ is } NP_{\Sigma^R} \text{-mach.} \land M(x) \downarrow\} \]

**UNI = RES-UNI** (the length of guesses can be restricted)

- Decompose $x_1, \ldots, x_n$ into equivalence classes,
- replace $x_1, \ldots, x_n$ by suitable short strings

such that possible chains are not destroyed.

**SUB-UNI** (short input strings) $\subseteq$ RES-UNI

- Transform the input tuple into a string,
- double the length,
- check the new string by means of $R$.

**Output:** $a / b$
Proof of $P_{\Sigma^R} = NP_{\Sigma^R}$ by a reduction.

$UNI = \{(x_1, \ldots, x_n, \text{Code}(M), b, \ldots, b) \mid x \in (U^*)^\infty \quad \& \quad M \text{ is } NP_{\Sigma^R}\text{-mach.} \quad \& \quad M(x)\downarrow\}$

**UNI = RES-UNI** (the length of guesses can be restricted)

- Decompose $x_1, \ldots, x_n$ into equivalence classes,
- replace $x_1, \ldots, x_n$ by suitable short strings

such that possible chains are not destroyed.

**SUB-UNI** (short input strings)

- Transform the input tuple into a string,
- double the length,
- check the new string by means of $R$.

Output: $a / b$
\[ P_{\Sigma_R} = NP_{\Sigma_R} \]

Proof of \( P_{\Sigma_R} = NP_{\Sigma_R} \) by a reduction.

\[ \text{UNI} = \{(x_1, \ldots, x_n, \text{Code}(M), b, \ldots, b) \mid x \in (U^*)^\infty \quad \& \quad M \text{ is NP}_{\Sigma_R} \text{-mach.} \quad \& \quad M(x) \downarrow \} \]

\[ \text{UNI} = \text{RES-UNI} \quad (\text{the length of guesses can be restricted}) \]

- Decompose \( x_1, \ldots, x_n \) into equivalence classes,
- replace \( x_1, \ldots, x_n \) by suitable short strings

such that possible chains are not destroyed.

\[ \text{SUB-UNI} \quad (\text{short input strings}) \quad \text{SUB-UNI} \subset \text{RES-UNI} \]

- Transform the input tuple into a string,
- double the length,
- check the new string by means of \( R \).

Output: \( a \ / \ b \)
The uniform model of computation
Oracles implying $P^A = NP^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$

Summary

Proof of $P_{\Sigma_R} = NP_{\Sigma_R}$ by a reduction.

$\text{UNI} = \{(x_1, \ldots, x_n, \text{Code}(M), b, \ldots, b) \mid x \in (U^*)^\infty \land M \text{ is NP}_{\Sigma_R}\text{-mach.} \land M(x) \downarrow\}$

$\text{UNI} = \text{RES-UNI}$ (the length of guesses can be restricted)

- Decompose $x_1, \ldots, x_n$ into equivalence classes,
- replace $x_1, \ldots, x_n$ by suitable short strings

such that possible chains are not destroyed.

$\text{SUB-UNI}$ (short input strings) $\subseteq \text{RES-UNI}$

- Transform the input tuple into a string,
- double the length,
- check the new string by means of $R$.

Output: $a / b$
The replacements for an ordered structure (1)

\((\mathbb{N}; 0, 1; \text{shr}, \text{shl}, \text{inc} \circ \text{shl}; R, \leq)\)

The binary code of an input: \[11000 \cdots 110101011011111011111111\]

Results after 9 steps:
\[11000 \cdots 110101011011111011111111\]
\[11000 \cdots 110101011011111011111100\]
\[11000 \cdots 110101011011111011111110\]
\[11000 \cdots 110101011011111011111110\]

Description by a tree:

The uniform model of computation
Oracles implying \(P^A = NP^A\)
Relations derived from oracles with \(P = NP\)
An ordered structure with \(P = NP\)

Summary
The replacements for an ordered structure (2)

\((\mathbb{N}; 0, 1; \text{shr}, \text{shl}, \text{inc} \circ \text{shl}; R, \leq)\)

If \(s\) satisfies \(R\), then \(s\) can be replaced by some \(s_0\)

\[
\begin{align*}
\text{bin}(s) &= r1^{\lfloor r \rfloor} \\
r &= \langle \text{bin}(x_1),\ldots,\text{bin}(x_n) \rangle \quad \text{Code}^*(M) \ 0^t \\
\text{bin}(s_0) &= r_01^{\lfloor r_0 \rfloor} \\
r_0 &= \langle 10\cdots \rangle \quad \text{Code}^*(M_0) \ 0000
\end{align*}
\]

any length \quad where \(\forall x \ M_0(x) \downarrow^4\)
The replacements for an ordered structure (3)

Long prefixes in the binary code of
- large guesses
- large inputs

can be replaced by short prefixes
- without changing the computation path
- such that the order remains valid.

\[
\begin{align*}
\text{long prefix } \bin(p_1) & \quad \rightarrow \quad \text{short prefix } \bin(q_1) \\
p_1 & \leq p_2 \\
q_1 & \leq q_2 \\
\text{long prefix } \bin(p_2) & \quad \rightarrow \quad \text{short prefix } \bin(q_2)
\end{align*}
\]
The uniform model of computation
Oracles implying \( P^A = N^A \)
Relations derived from oracles with \( P = NP \)
An ordered structure with \( P = NP \)

**Summary**

\( P = NP \)

for

\[ \Sigma = (\mathbb{N}; 0, 1; \text{shr}, \text{shl}, \text{inc} \circ \text{shl}; R, \leq) \]

The reduction (Proof of \( P_\Sigma = NP_\Sigma \)):

\[ \text{UNI}_\Sigma = \text{RES-UNI}_\Sigma \]

\( \text{the guesses can be restricted such that the order remains valid} \)

Replace inputs such that the order remains valid.

\[ \text{SUB-UNI}_\Sigma \]

\( \text{(small inputs)} \)

\[ \text{SUB-UNI}_\Sigma \subset \text{RES-UNI}_\Sigma \]

Output: \( a / b \)
A summary of the ideas for the construction of structures of finite signature with $P = NP$

Summary of ideas:

1. Koiran (1994): \(\text{DNP} = \text{NP}\) for \((\mathbb{R}; 0, 1; +, -; \leq)\)
3. Prunescu (2001): Talk on an idea to define an additional relation over a term algebra which implies $P = NP$
5. Gaßner (2004): $P = NP$ for structures over trees with decidable identity

Replace arbitrary guesses by small guesses

Define an additional relation

Derive the new relations from a universal oracle

Slow operations for a recursive definition of a relation

Padding the codes of the members of a universal oracle

Several investigations of computation paths

The uniform model of computation
Oracles implying $P^A = NP^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$

Summary
Several structures of finite signature with $P = NP$

- An additional relation $R$ on padded codes of the members of a universal oracle

  - Binary trees with decidable identity relation
  - Trees of linear width with identity relation
  - Strings with operations for adding and deleting the last character
  - An ordered structure over integers

The uniform model of computation
Oracles implying $P^A = NP^A$
Relations derived from oracles with $P = NP$
An ordered structure with $P = NP$

Summary
In order to get \( P_{\Sigma_R} = NP_{\Sigma_R} \) we can choose:

- **Using some oracle** as \( O_\Sigma \) and padding the elements.
  The oracle can be derived from:
  - \( UNI_\Sigma \)
  - \( SAT_\Sigma \)

- **A directly recursive definition of the relation** \( R \)
  (analogously to an oracle).

Outlook:
Which form of definitions of an additional relation are possible?

The uniform model of computation
Oracles implying \( P^A = NP^A \)
Relations derived from oracles with \( P = NP \)
An ordered structure with \( P = NP \)

Summary
Some open questions

- Does there hold $P = NP$ for some known structures of finite signature?

- Are there other constructions of structures with $P = NP$ without using padding?
Some open questions

Does there hold $P = NP$ for some known structures of finite signature?

Are there other constructions of structures with $P = NP$ without using padding?

The uniform model of computation

Oracles implying $P^A = NP^A$

Relations derived from oracles with $P = NP$

An ordered structure with $P = NP$

Summary
Thank you for your attention!

Christine Gaßner
Greifswald.

Thanks also to
Volkmar Liebscher,
Rainer Schimming.
Appendix: The new relation $R$ (some details)

$\Sigma = (U^*; \varepsilon, a, b, c_3, \ldots, c_u; \text{add, sub}_1, \text{sub}_r, f_1, \ldots, f_v; R_1, \ldots, R_w, =)$

$\Sigma_R$ = Expansion of $\Sigma$ by $R$

$W_\Sigma^{(0)} = \emptyset$

$W_\Sigma^{(i+1)} = \{([\infty], \text{Code}(M), [b^i]) \in U^i \mid M \text{ is non-det. } W_{\Sigma}^{(i)}\text{-mach.} \& M(x)\downarrow \}$

$W_\Sigma = \bigcup_{i \geq 1} W_\Sigma^{(i)} \quad \Rightarrow \quad P_{\Sigma}^{W_\Sigma} = \text{NP}_{\Sigma}^{W_\Sigma}$

$s = d_1 \cdots d_k \in U^*$

$[d_1 \cdots d_k] = (d_1, \ldots, d_k) \in U^k$

$x \in (U^*)^\infty$

$\infty \in U^*$

The relation $R$:

$R(s) \iff (\exists r \in U^*) ([r] \in W_\Sigma \& s = ra^{|r|})$

Theorem:

$P_{\Sigma_R} = \text{NP}_{\Sigma_R}$