Towards a Formal Proof of the Cook-Levin Theorem

Lennard Gäher
Advisor: Fabian Kunze
Supervisor: Prof. Smolka

Saarland University

15 April 2020

Final Bachelor Talk

The Cook-Levin Theorem

SAT is NP-complete [Cook, 1971, Levin, 1973].

SAT

Given a Boolean formula in conjunctive normal form, does there exist a satisfying assignment?

[Karp, 1972]: 21 more combinatorial NP-complete problems

The Cook-Levin Theorem

SAT is NP-complete [Cook, 1971, Levin, 1973].

SAT

Given a Boolean formula in conjunctive normal form, does there exist a satisfying assignment?

[Karp, 1972]: 21 more combinatorial NP-complete problems

Mechanisation [Gamboa and Cowles, 2004]: verified construction without connection to computational model (ACL2)

Our Computational Model

Prevalent computational model: Turing machines

> *Turing machines as model of computation are inherently infeasible for the formalisation of any computability- or complexity-theoretic result*
>
> [Forster et al., 2020]

Here: use λ-calculus L
Supported by:

- reasonability of L [Forster et al., 2019]
- extraction mechanism in Coq [Forster and Kunze, 2019]

size explosion: **"time bounds space" does not hold**

Basic Complexity Theory

Problems $P, Q : X \to \mathbb{P}$
NP: verifier characterisation

$$P \preceq_p Q := \exists f : X \to Y, \quad (\forall x, P\ x \leftrightarrow Q(f\ x))$$
$$\wedge \text{ polynomial-time computable } f$$
$$\wedge\ f \text{ increases the size polynomially}$$

$$\text{NP-hard } P := \forall Q, Q \in \text{NP} \to Q \preceq_p P$$
$$\text{NP-complete } P := P \in \text{NP} \wedge \text{NP-hard } P$$

Proving the Cook-Levin Theorem

SAT is NP-complete.

SAT

Given a Boolean formula in conjunctive normal form, does there exist a satisfying assignment?

Idea: encode computation as Boolean formula

L: non-local computations, too high-level ☹

Proving the Cook-Levin Theorem

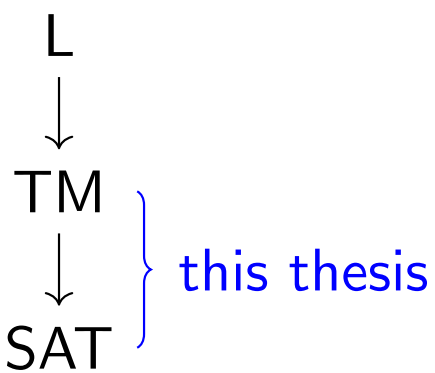
SAT is NP-complete.

SAT

Given a Boolean formula in conjunctive normal form, does there exist a satisfying assignment?

Idea: encode computation as Boolean formula

L: non-local computations, too high-level ☹



Generic Problem for Turing Machines

- formalisation by [Asperti and Ricciotti, 2015] and mechanisation from [Forster et al., 2020]
- two-sided infinite tapes
- no explicit blanks

Generic Problem for Turing Machines

- formalisation by [Asperti and Ricciotti, 2015] and mechanisation from [Forster et al., 2020]
- two-sided infinite tapes
- no explicit blanks

TMGenNP

$$\textbf{TMGenNP}\ (M, \mathit{inp}, t) := M \text{ is a } \mathit{nondet.} \text{ 1-tape TM}$$
$$\wedge M \text{ accepts on } \mathit{inp} \text{ in } \leq t \text{ steps}$$

Generic Problem for Turing Machines

- formalisation by [Asperti and Ricciotti, 2015] and mechanisation from [Forster et al., 2020]
- two-sided infinite tapes
- no explicit blanks

TMGenNP

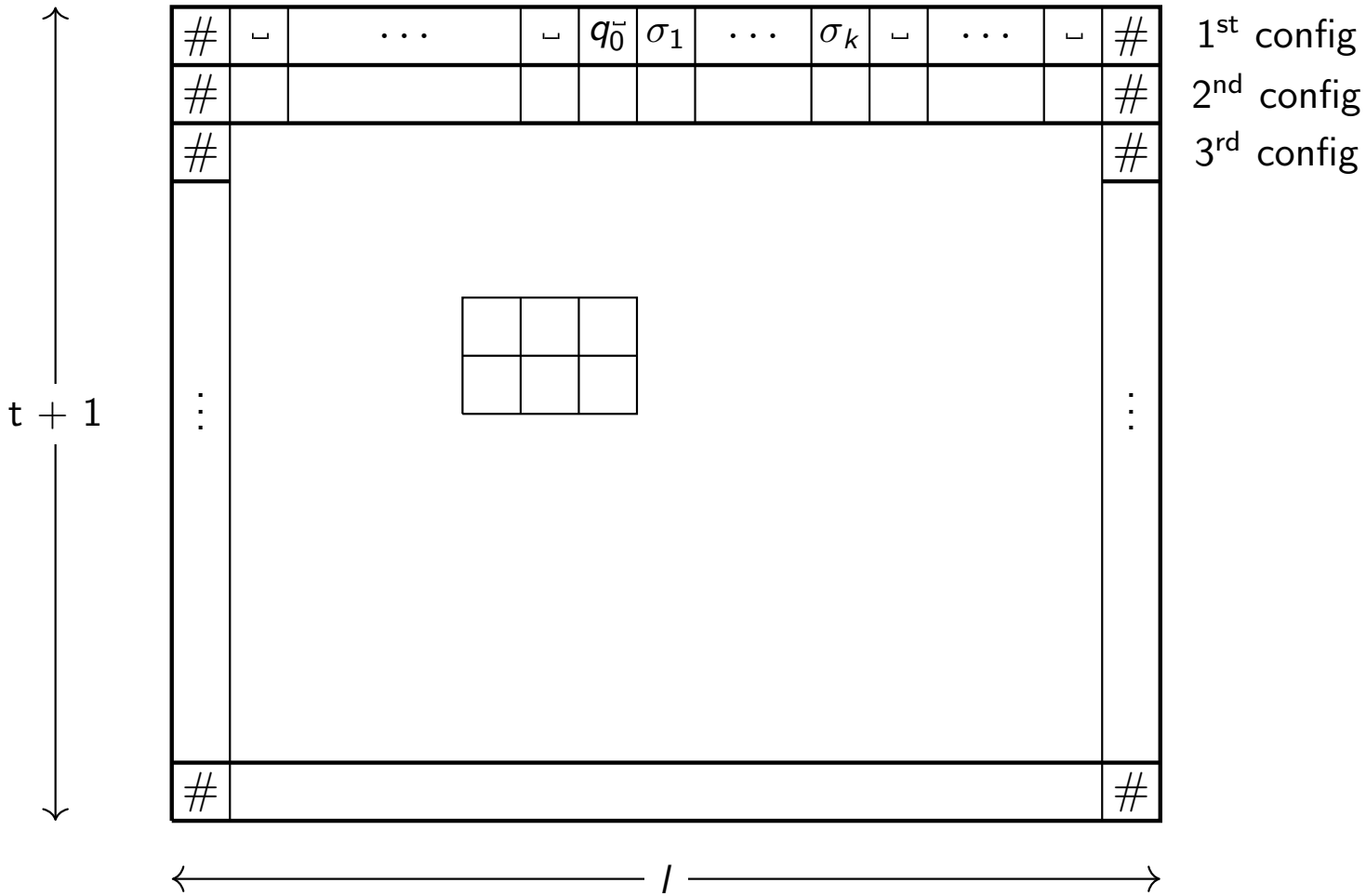
$$\begin{aligned}\textbf{TMGenNP}\ (M, \mathit{inp}, k', t) := {} & M \text{ is a } \mathit{det.} \text{ 1-tape TM} \\ \wedge & \exists\ \mathit{cert}, |\mathit{cert}| \leq k' \\ \wedge & M \text{ accepts on } \mathit{inp} +\!\!+ \mathit{cert} \text{ in } \leq t \text{ steps}\end{aligned}$$

Bounded Size

SAT formula has a fixed size, but:

- TM may have different space usage depending on input
- TM may take a different number of steps until it halts

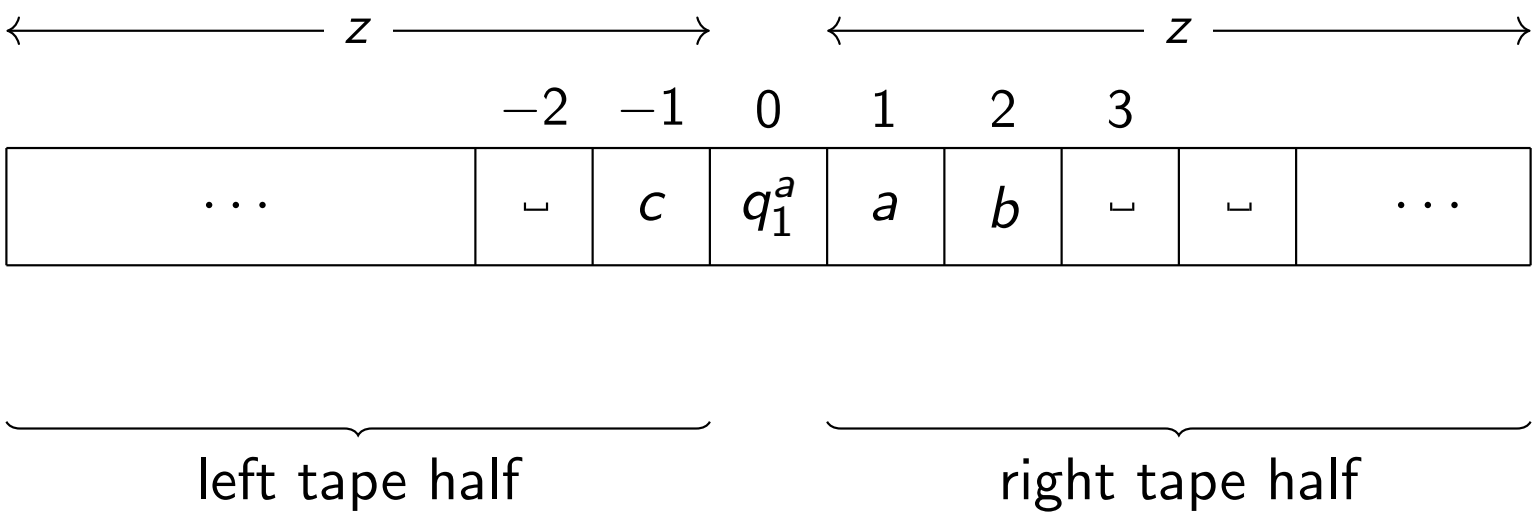
Tableau of Configurations[1]



[1] based on [Sipser, 1997], similar to [Cook, 1971]

Configuration String

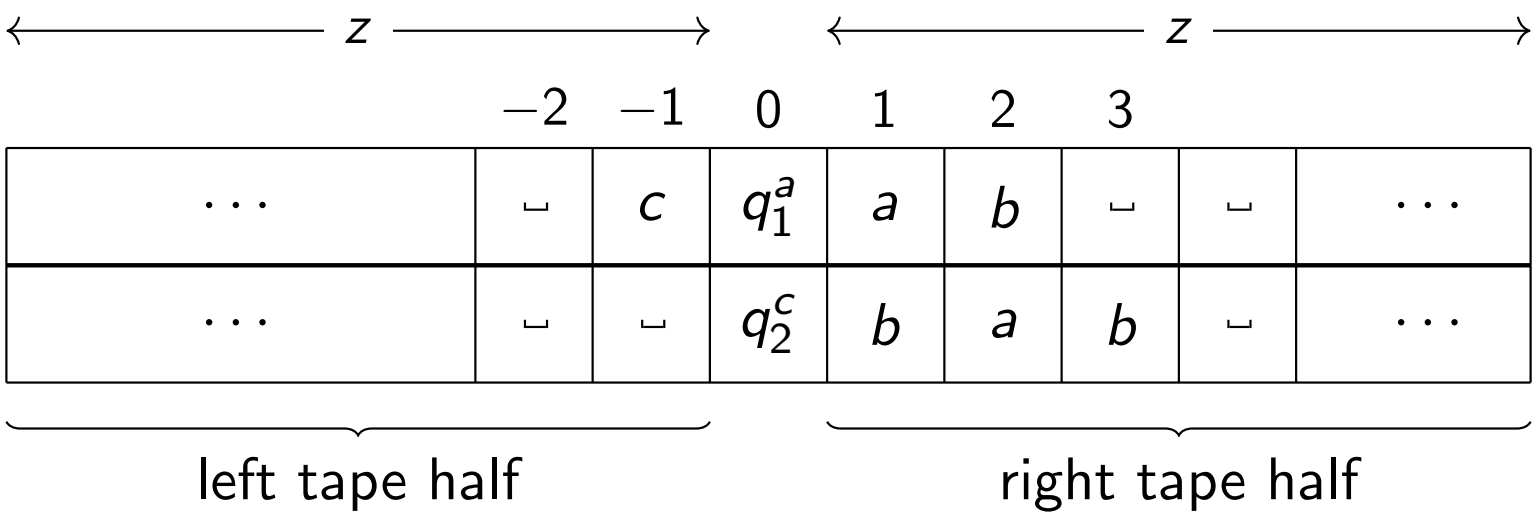
$\Sigma_{\mathsf{TM}} = \{a, b, c\}$



special blanks ␣ for unused regions of the string

Configuration String

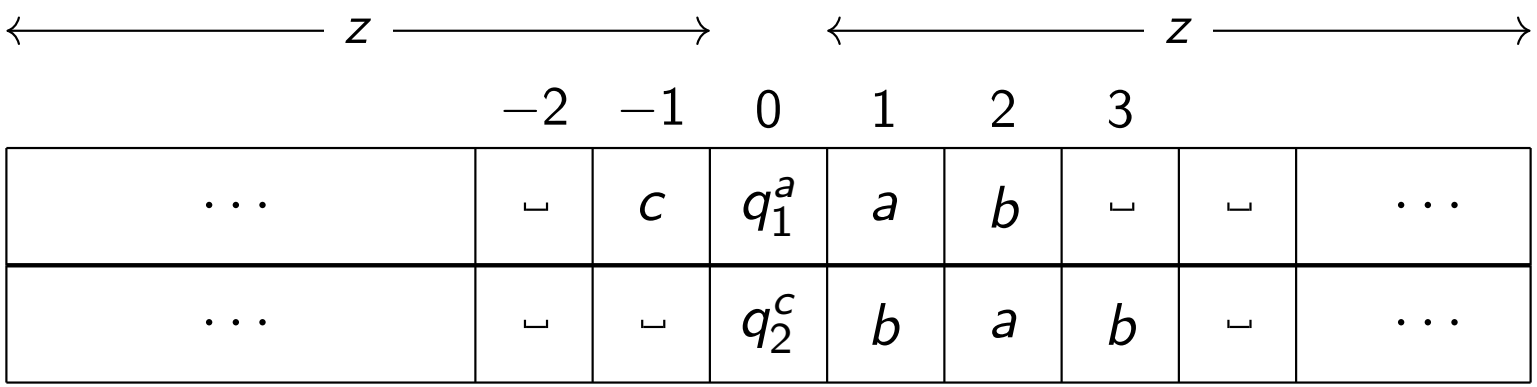
$\Sigma_{\mathsf{TM}} = \{a, b, c\}$ $\qquad \delta(q_1, a) = (q_2, {}^{\circ}b, \mathsf{L})$



special blanks ␣ for unused regions of the string

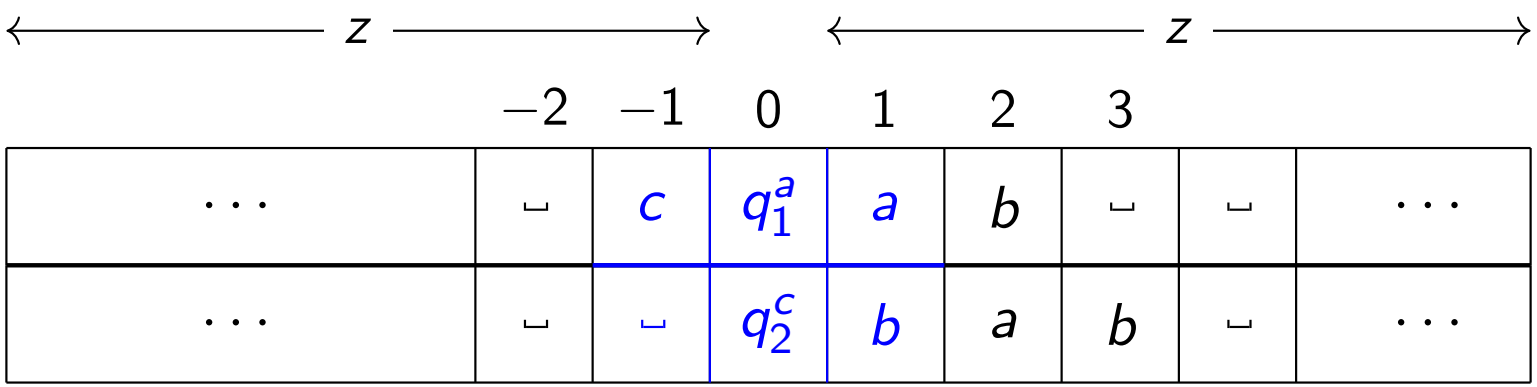
Rewrite Windows: Force Valid Configuration Changes

$\Sigma_{\mathsf{TM}} = \{a, b, c\}$ $\qquad$ $\delta(q_1, a) = (q_2, {}^{\circ}b, \mathsf{L})$



Rewrite Windows: Force Valid Configuration Changes

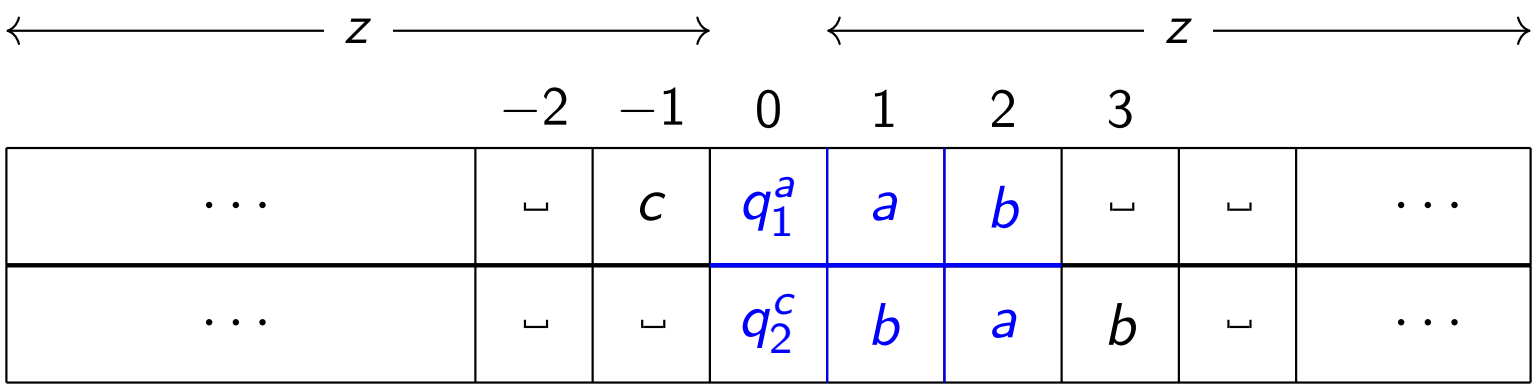
$\Sigma_{\mathsf{TM}} = \{a, b, c\}$ $\qquad$ $\delta(q_1, a) = (q_2, {}^{\circ}b, \mathrm{L})$



c	q_1^a	a
␣	q_2^c	b

Rewrite Windows: Force Valid Configuration Changes

$\Sigma_{\mathsf{TM}} = \{a, b, c\}$ $\qquad \delta(q_1, a) = (q_2, {}^{\circ}b, \mathsf{L})$

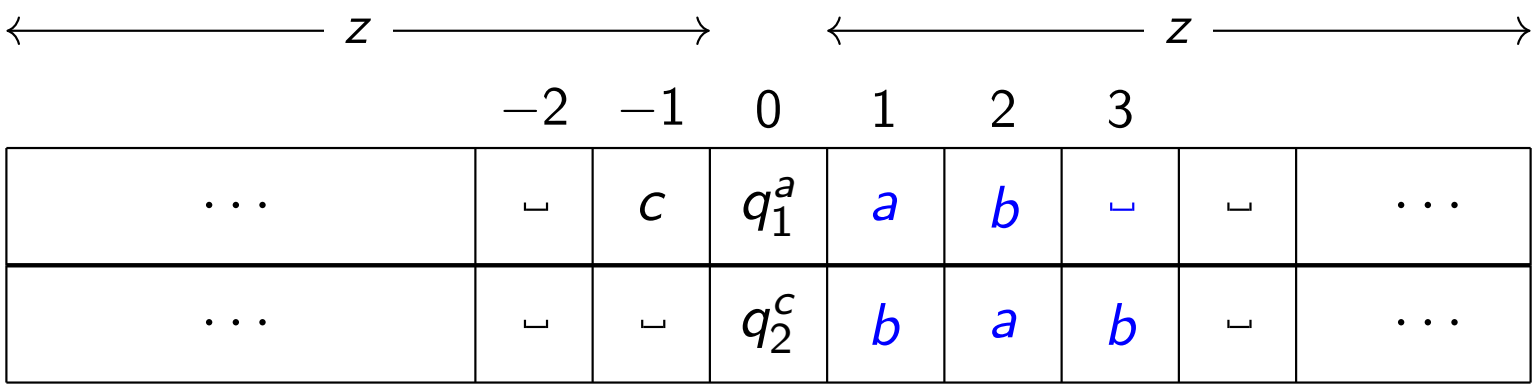


c	q_1^a	a
␣	q_2^c	b

q_1^a	a	b
q_2^c	b	a

Rewrite Windows: Force Valid Configuration Changes

$\Sigma_{\mathsf{TM}} = \{a, b, c\}$ $\qquad$ $\delta(q_1, a) = (q_2, {}^{\circ}b, \mathrm{L})$



c	q_1^a	a
␣	q_2^c	b

q_1^a	a	b
q_2^c	b	a

a	b	␣
b	a	b

Parallel Rewriting (**PR**)

Given:

- an initial string $x_0 \in \Sigma^l$ over an alphabet Σ
  representation of initial config
- a set of rewrite windows R of width w
  possible local behaviours of the Turing machine
- a step count t
  number of TM steps
- a set of final substring constraints R_{final}
  symbols of accepting states

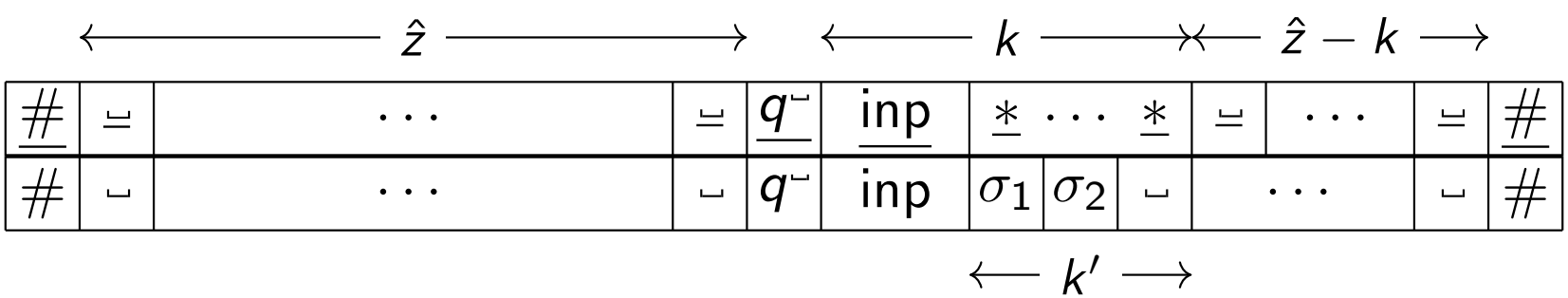
Determine: $\exists\, x_1, \ldots, x_t \in \Sigma^l$ s.t.

- $x_i \rightsquigarrow x_{i+1}$: "for all offsets there exists a rewrite window"
- there exists an element $x \in R_{final}$ which is a substring of x_t

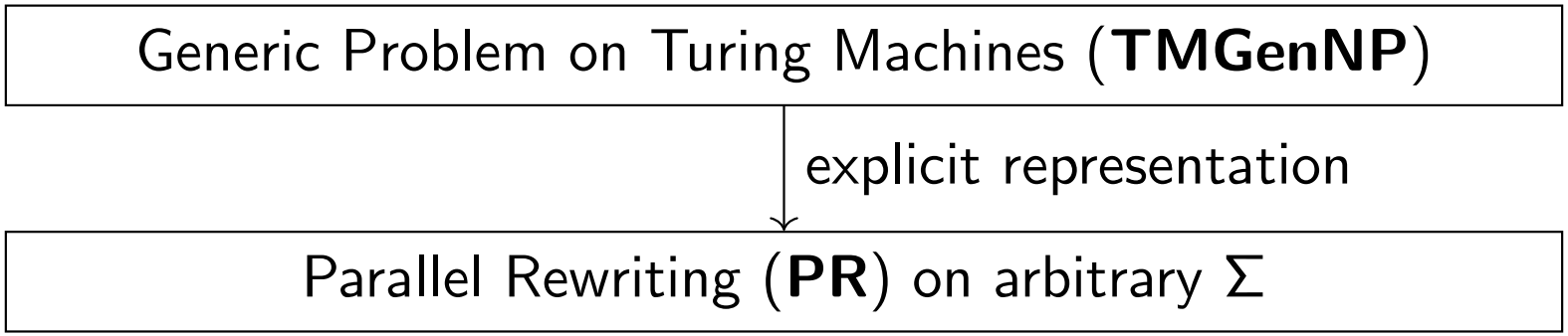
Nondeterministic Certificate

- "guess" certificate of length $\leq k'$ with a single rewrite step
- add symbols $\{\underline{\#}, \underline{\sqcup}, \underline{*}, \underline{q^{\sqcup}}, \underline{\sigma}\}$ for initial state q and $\sigma : \Sigma_{\mathsf{TM}}$

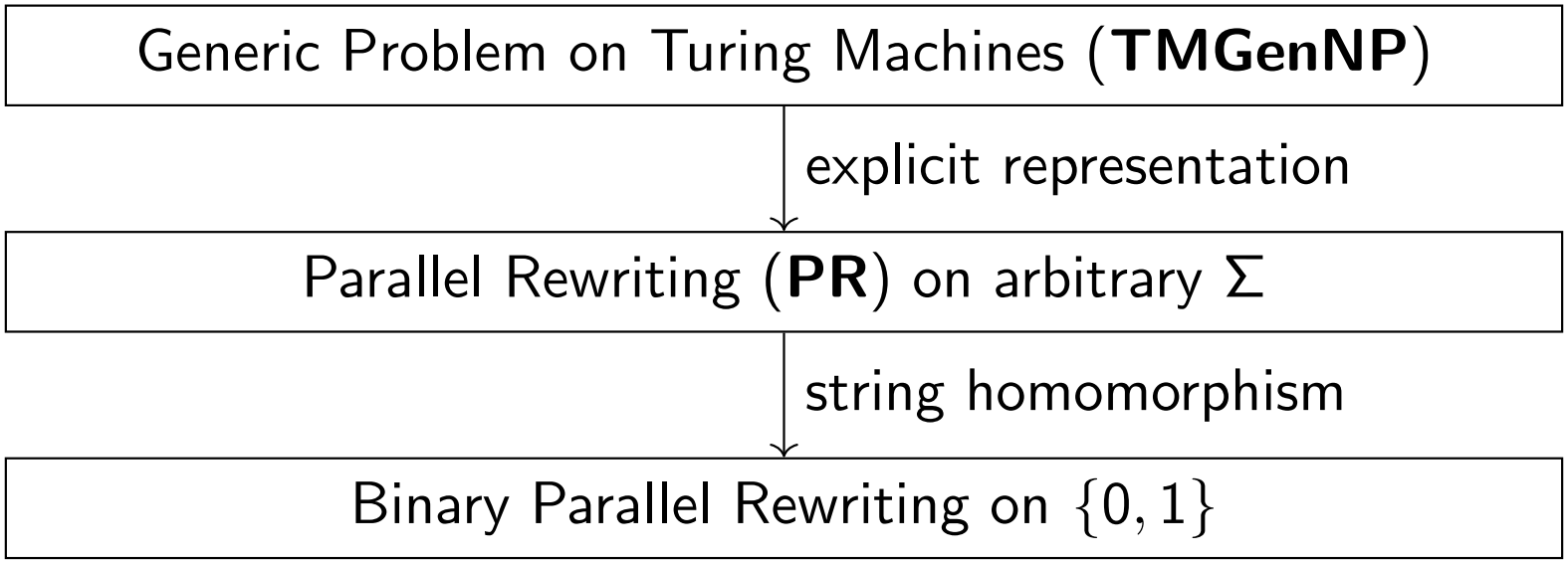
Initial string:



Chain of Reductions



Chain of Reductions



To a Binary Alphabet

- arbitrary alphabet $\Sigma = \{\sigma_0, \ldots, \sigma_{|\Sigma|-1}\}$
- string homomorphism $h : \Sigma^* \to \{0,1\}^*$

$$\sigma_i \mapsto 0^i 10^{|\Sigma|-i-1}$$

- in general: injective *uniform* homomorphisms
  $\to$ scale length up by constant factor

Example ($\Sigma = \{a, b\}$):

$$a \mapsto 01$$
$$b \mapsto 10$$

To a Binary Alphabet

- arbitrary alphabet $\Sigma = \{\sigma_0, \ldots, \sigma_{|\Sigma|-1}\}$
- string homomorphism $h : \Sigma^* \to \{0,1\}^*$

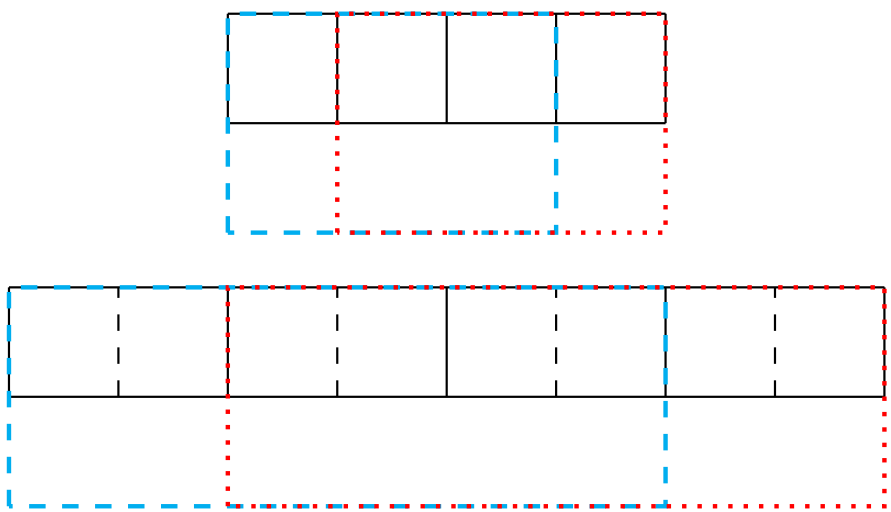
$$\sigma_i \mapsto 0^i 10^{|\Sigma|-i-1}$$

- in general: injective *uniform* homomorphisms
  $\to$ scale length up by constant factor

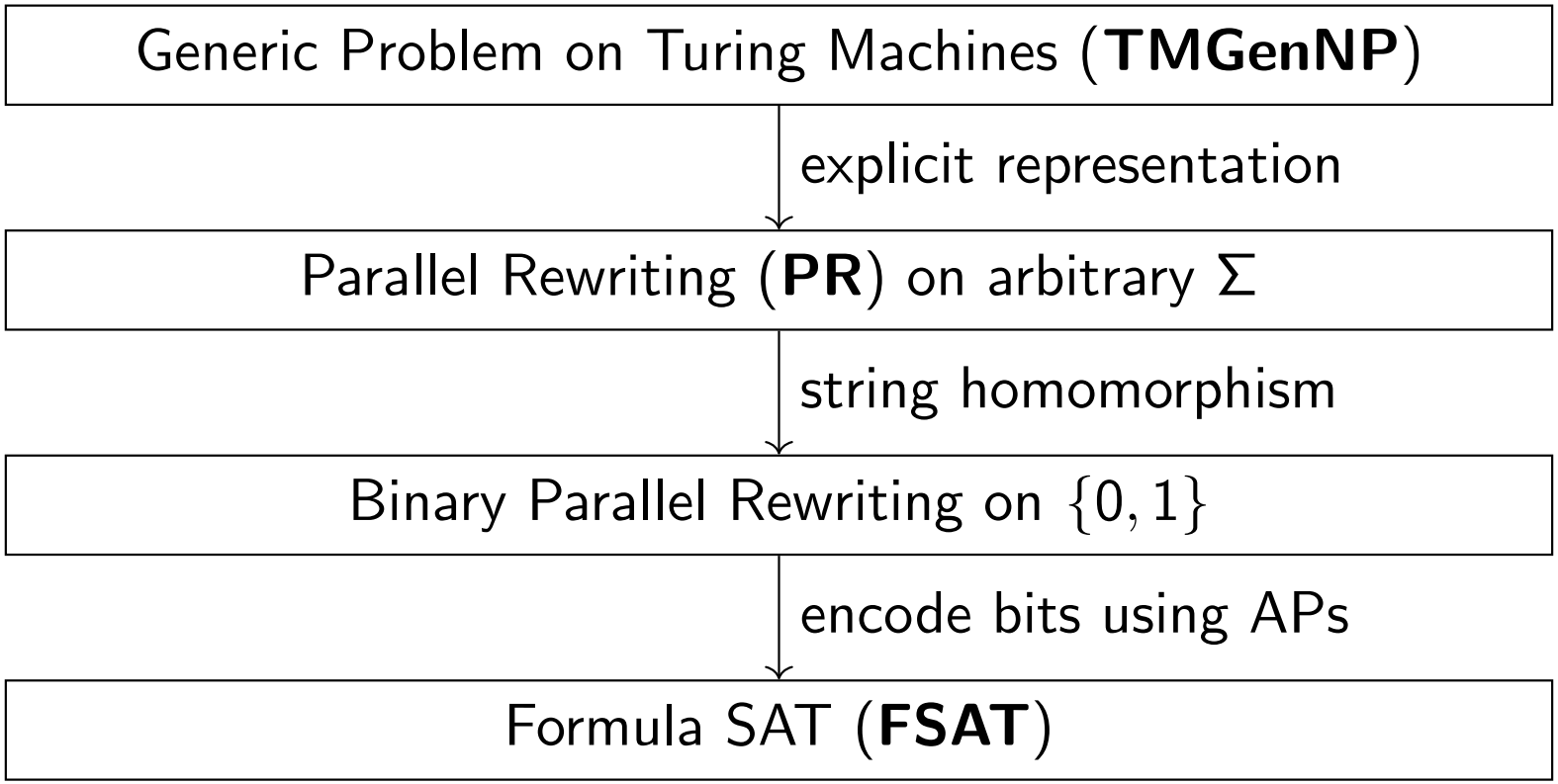
Example ($\Sigma = \{a, b\}$):

$a \mapsto 01$

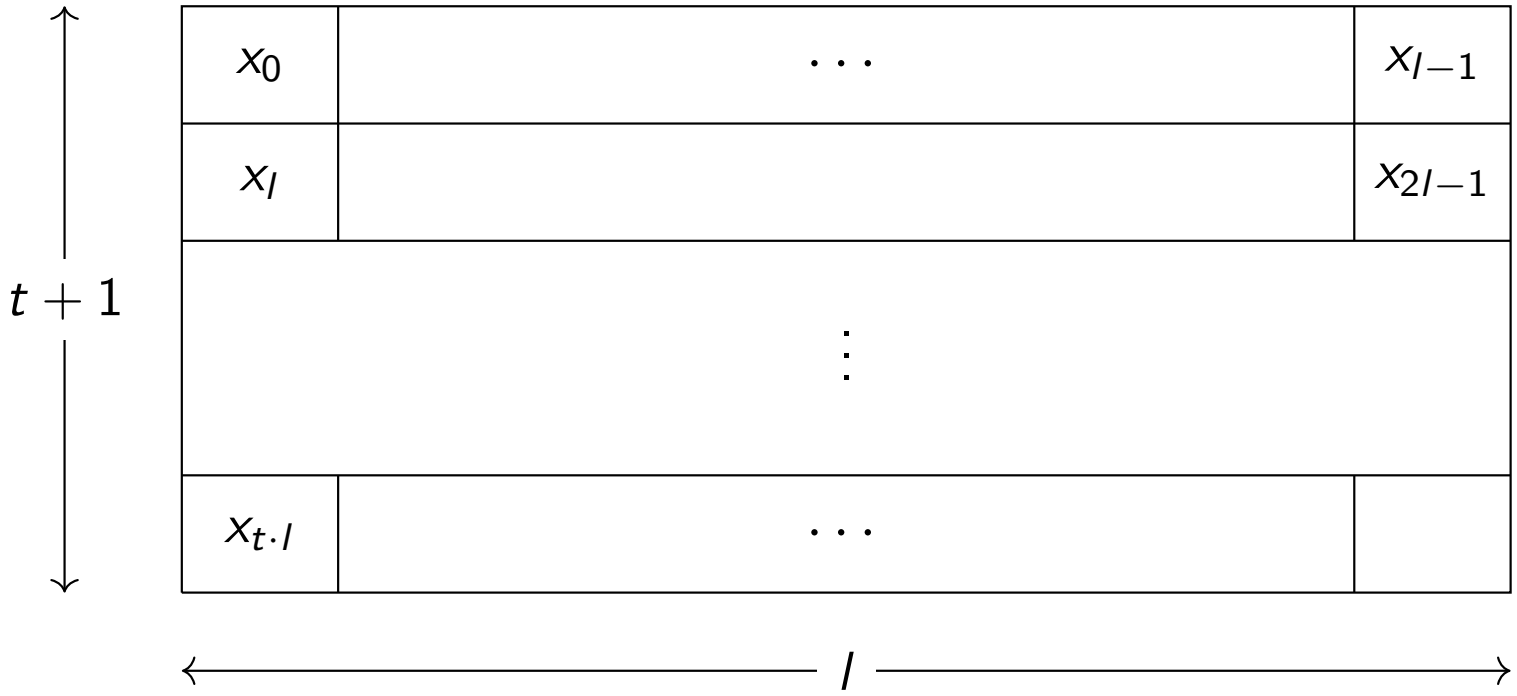
$b \mapsto 10$



Chain of Reductions



Encoding as a Boolean Formula

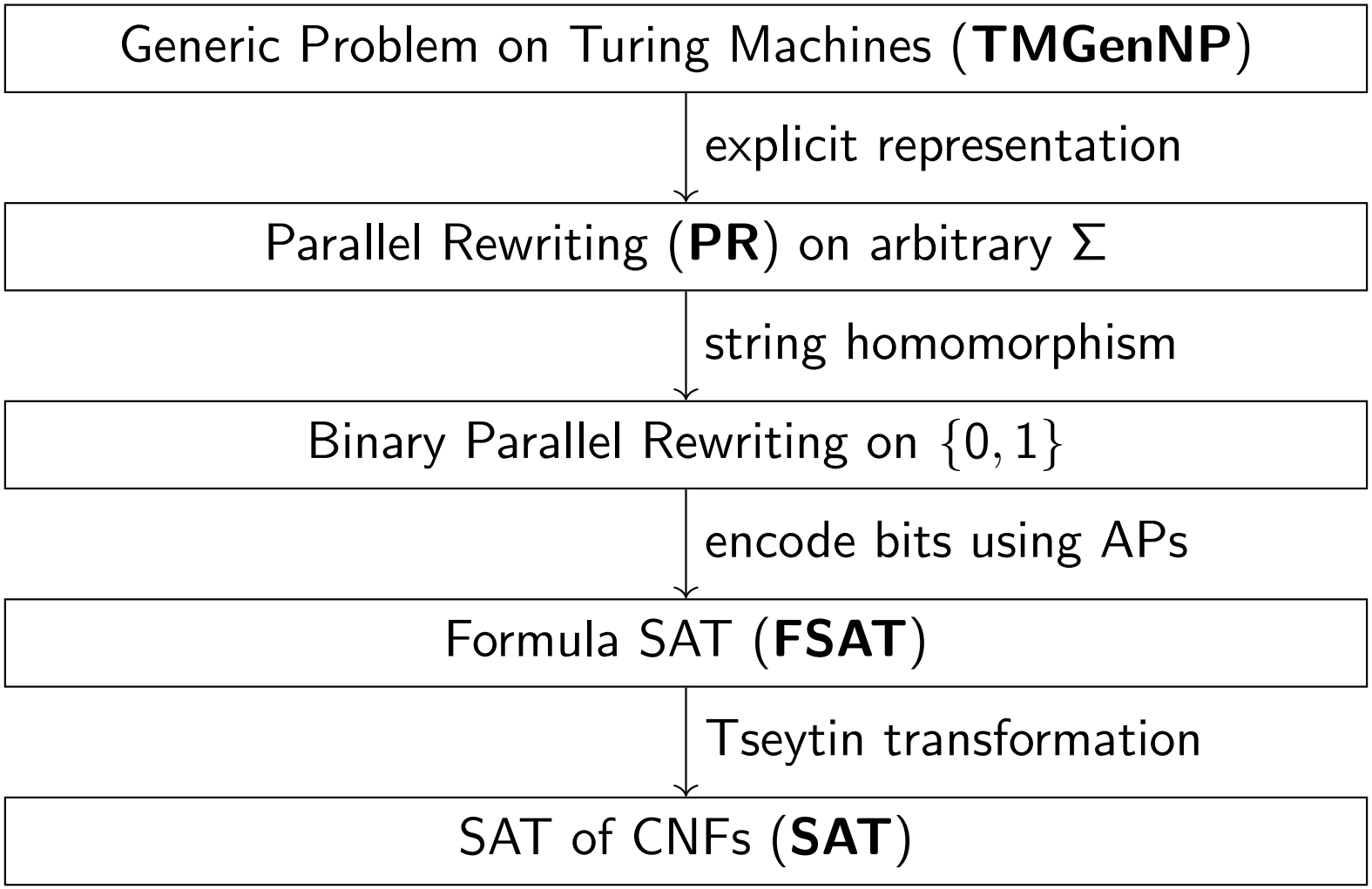


$$\Phi := \Phi_{\text{init}} \wedge \Phi_{\text{rewrite}} \wedge \Phi_{\text{final}}$$

$$\Phi_{\text{rewrite}} := \bigwedge_{\text{rewrites}} \bigwedge_{\text{offsets}} \bigvee_{w \in R} \bigwedge_{\text{characters of } w} \text{encodeBit}$$

$x_i \rightsquigarrow x_{i+1}$: "for all offsets there exists a rewrite window"

Chain of Reductions



Contributions

Main result

If **TMGenNP** is NP-hard, then **SAT** is NP-complete.

- factorisation of the textbook proof [Sipser, 1997] to make mechanisation feasible: new **PR** problem
- full running-time analysis in L

Moreover:

- **TMGenNP** $\in$ NP
- reduction k-**SAT** $\preceq_p$ **Clique**
- conditional NP-completeness of **Clique**
- formalisation of $\#P$

Future Work

- missing reduction from L to **TMGenNP**
- space-related results, e.g. Savitch
- binary encodings

LOC

Component		Spec	Proof
complexity definitions by Fabian		290	506
preliminaries		281	634
flat finite types		83	223
problem definitions + NP proofs		1093	1860
TMGenNP to **PR**	correctness	1843	2481
	time analysis	838	1706
3-PR to **PR**		37	174
PR to **BinaryPR**		222	719
BinaryPR to **FSAT**		312	1078
FSAT to **SAT**		213	605
k-**SAT** to **Clique**		386	773
total		5230	10038
without extraction		3594	6857
without computability		2339	4524

Comparison with [Gamboa and Cowles, 2004]

- existing formalisation in ACL2
- differences in setting:
  - single-sided infinite tapes
  - explicit nondeterminism instead of verifiers
  - reduction only to **FSAT**
- *direct* reduction, no factorisation $\rightarrow$ only seems to be feasible because of restricted setting
- no running time analysis with respect to reasonable model

Rewrite Rules

Add one symbol to the right half of the tape:

a	b	$\sqcup$	$\sqcup$	$\cdots$
b	a	b	$\sqcup$	$\cdots$

a	b	$\sqcup$
b	a	b

Rewrite Rules

Add one symbol to the right half of the tape:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
σ_3	σ_1	σ_2	$\sqcup$	$\cdots$

$$\sigma_i \in \Sigma_{\mathsf{TM}}$$

σ_1	σ_2	$\sqcup$
σ_3	σ_1	σ_2

Rewrite Rules

Add one symbol to the right half of the tape:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
σ_3	σ_1	σ_2	$\sqcup$	$\cdots$

$$\sigma_i \in \Sigma_{\mathsf{TM}}$$

σ_1	σ_2	$\sqcup$
σ_3	σ_1	σ_2

σ_2	$\sqcup$	$\sqcup$
σ_1	σ_2	$\sqcup$

$\sqcup$	$\sqcup$	$\sqcup$
σ_2	$\sqcup$	$\sqcup$

$\cdots$

Tape Shifts

Add one symbol to the right half of the tape:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
σ_3	σ_1	σ_2	$\sqcup$	$\cdots$

σ_1	σ_2	$\sqcup$
σ_3	σ_1	σ_2

Tape Shifts

Add one symbol to the right half of the tape:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
σ_3	σ_1	σ_2	$\sqcup$	$\cdots$

σ_1	σ_2	$\sqcup$
σ_3	σ_1	σ_2

$\sigma_i \in \Sigma_{\mathsf{TM}}$

Tape Shifts

Add one symbol to the right half of the tape:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
σ_3	σ_1	σ_2	$\sqcup$	$\cdots$

σ_1	σ_2	$\sqcup$
σ_3	σ_1	σ_2

$\sigma_i \in \Sigma_{\mathsf{TM}}$

Leave the tape unchanged:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$

σ_1	σ_2	$\sqcup$
σ_1	σ_2	$\sqcup$

Tape Shifts

Add one symbol to the right half of the tape:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
σ_3	σ_1	σ_2	$\sqcup$	$\cdots$

σ_1	σ_2	$\sqcup$
σ_3	σ_1	σ_2

$\sigma_i \in \Sigma_{\mathsf{TM}}$

Leave the tape unchanged:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$

σ_1	σ_2	$\sqcup$
σ_1	σ_2	$\sqcup$

$\cdots$	$\sqcup$	c	q_1^a	b	b	$\sqcup$	$\sqcup$	$\cdots$
$\cdots$	$\sqcup$	$\sqcup$	q_2^c	b	b	b	$\sqcup$	$\cdots$

Tape Shifts

Add one symbol to the right half of the tape:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
σ_3	σ_1	σ_2	$\sqcup$	$\cdots$

σ_1	σ_2	$\sqcup$
σ_3	σ_1	σ_2

$\sigma_i \in \Sigma_{\mathsf{TM}}$

Leave the tape unchanged:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$

σ_1	σ_2	$\sqcup$
σ_1	σ_2	$\sqcup$

$\cdots$	$\sqcup$	c	q_1^a	b	b	$\sqcup$	$\sqcup$	$\cdots$
$\cdots$	$\sqcup$	$\sqcup$	q_2^c	b	b	$\sqcup$	$\sqcup$	$\cdots$

Tape Shifts

Add one symbol to the right half of the tape:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
σ_3	σ_1	σ_2	$\sqcup$	$\cdots$

σ_1	σ_2	$\sqcup$
σ_3	σ_1	σ_2

$\sigma_i \in \Sigma_{\mathsf{TM}}$

Leave the tape unchanged:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$

σ_1	σ_2	$\sqcup$
σ_1	σ_2	$\sqcup$

$\cdots$	$\sqcup$	c	q_1^a	b	b	$\sqcup$	$\sqcup$	$\cdots$
$\cdots$	$\sqcup$	$\sqcup$	q_2^c	b	b	$\sqcup$	$\sqcup$	$\cdots$

Polarities $\{\overleftarrow{\cdot}, \overline{\cdot}, \overrightarrow{\cdot}\}$

Add one symbol to the right half of the tape:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
$\overrightarrow{\sigma_3}$	$\overrightarrow{\sigma_1}$	$\overrightarrow{\sigma_2}$	$\sqcup$	$\cdots$

σ_1	σ_2	$\sqcup$
$\overrightarrow{\sigma_3}$	$\overrightarrow{\sigma_1}$	$\overrightarrow{\sigma_2}$

Leave the tape unchanged:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
$\overline{\sigma_1}$	$\overline{\sigma_2}$	$\sqcup$	$\sqcup$	$\cdots$

σ_1	σ_2	$\sqcup$
$\overline{\sigma_1}$	$\overline{\sigma_2}$	$\sqcup$

Polarities $\{\overleftarrow{\cdot}, \overline{\cdot}, \overrightarrow{\cdot}\}$

Add one symbol to the right half of the tape:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
$\overrightarrow{\sigma_3}$	$\overrightarrow{\sigma_1}$	$\overrightarrow{\sigma_2}$	$\sqcup$	$\cdots$

σ_1	σ_2	$\sqcup$
$\overrightarrow{\sigma_3}$	$\overrightarrow{\sigma_1}$	$\overrightarrow{\sigma_2}$

Leave the tape unchanged:

σ_1	σ_2	$\sqcup$	$\sqcup$	$\cdots$
$\overline{\sigma_1}$	$\overline{\sigma_2}$	$\sqcup$	$\sqcup$	$\cdots$

σ_1	σ_2	$\sqcup$
$\overline{\sigma_1}$	$\overline{\sigma_2}$	$\sqcup$

$\cdots$	$\sqcup$	c	q_1^a	b	b	$\sqcup$	$\sqcup$	$\cdots$
$\cdots$	$\sqcup$	$\sqcup$	q_2^c	$\overrightarrow{b}$	$\overrightarrow{b}$	$\overrightarrow{b}$	$\sqcup$	$\cdots$

Transition Rules – Example

$m \in \Sigma_{\mathsf{TM}} \cup \{\sqcup\}, \sigma \in \Sigma_{\mathsf{TM}}$

$\delta(q, a) = (p, {}^{\circ}b, L)$:

$\sqcup$	q^a	m_1
$\sqcup$	$p^{\sqcup}$	$\overrightarrow{b}$

σ_1	q^a	m_1
$\overrightarrow{m_2}$	p^{σ_1}	$\overrightarrow{b}$

$\sqcup$	$\sqcup$	q^a
$\sqcup$	$\sqcup$	$p^{\sqcup}$

$\sqcup$	σ_1	q^a
$\sqcup$	$\sqcup$	p^{σ_1}

σ_1	σ_2	q^a
$\overrightarrow{m_1}$	$\overrightarrow{\sigma_1}$	p^{σ_2}

q^a	$\sqcup$	$\sqcup$
p^{m_1}	$\overrightarrow{b}$	$\sqcup$

q^a	σ_1	m_1
p^{m_2}	$\overrightarrow{b}$	$\overrightarrow{\sigma_1}$

Transition Rules – Example

$m \in \Sigma_{\mathsf{TM}} \cup \{\sqcup\}, \sigma \in \Sigma_{\mathsf{TM}}$

In Coq mechanisation:
$\delta(q, a) = (p, {}^{\circ}b, L)$:

m_1	q^a	m_2
$\overrightarrow{m_3}$	p^{m_1}	$\overrightarrow{b}$

m_1	m_2	q^a
$\overrightarrow{m_3}$	$\overrightarrow{m_1}$	p^{m_2}

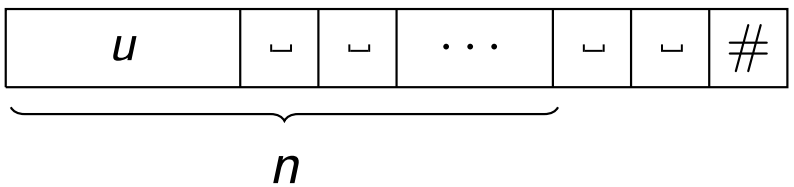
q^a	m_1	m_2
p^{m_3}	$\overrightarrow{b}$	$\overrightarrow{m_1}$

Induces garbage, e.g.

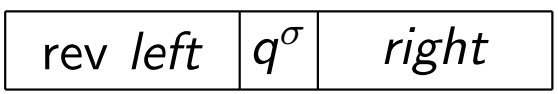
$\sqcup$	q^a	$\sqcup$
$\overrightarrow{\sigma}$	$p^{\sqcup}$	$\overrightarrow{b}$

Representation Relations

Representation of tape halves:

$u \sim_t^n$ 

Representation of configurations:

$q; (ls, \sigma, rs) \sim_c$ 

, where:

- $ls \sim_t^{z'} left$
- $rs \sim_t^{z'} right$

Deterministic Simulation

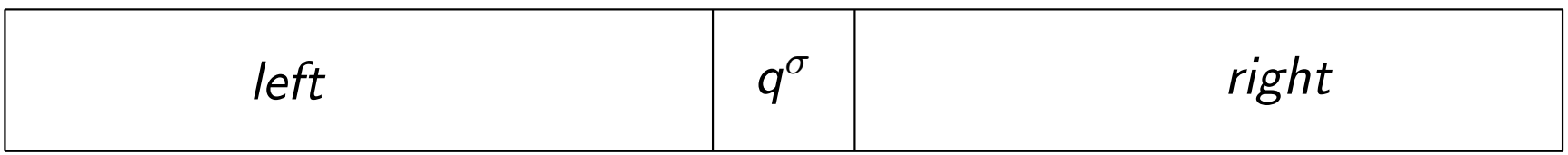
$$
\begin{array}{rcl}
q;(ls,\sigma,rs) & \sim_c & \boxed{\text{rev } left \mid q^{\sigma} \mid right} \\
\Upsilon & & \wr \\
q';(ls',\sigma',rs') & \sim_c & \boxed{\text{rev } left' \mid q'^{\sigma'} \mid right'}
\end{array}
$$

where $ls \sim_t^{z'} left, rs \sim_t^{z'} right$ and $ls' \sim_t^{z'} left', rs' \sim_t^{z'} right'$

Deterministic Simulation

$$
\begin{array}{lcl}
q;(ls,\sigma,rs) & \sim_c & \boxed{\text{rev } left \mid q^{\sigma} \mid right} \\
\Upsilon & & \wr \\
q';(ls',\sigma',rs') & \sim_c & \boxed{\text{rev } left' \mid q'^{\sigma'} \mid right'}
\end{array}
$$

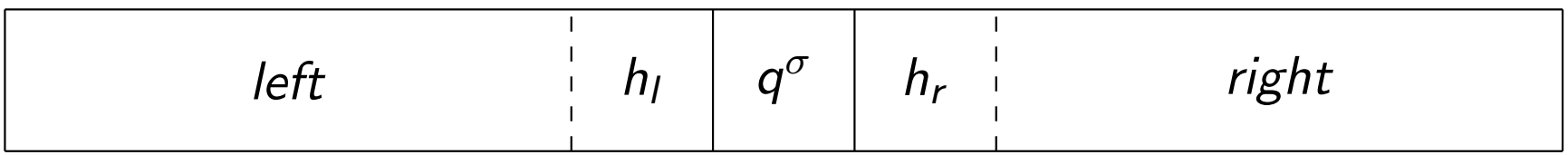
where $ls \sim_t^{z'} left, rs \sim_t^{z'} right$ and $ls' \sim_t^{z'} left', rs' \sim_t^{z'} right'$



Deterministic Simulation

$$
\begin{array}{lll}
q;(ls,\sigma,rs) & \sim_c & \boxed{\text{rev } left \mid q^{\sigma} \mid right} \\
\Upsilon & & \wr \\
q';(ls',\sigma',rs') & \sim_c & \boxed{\text{rev } left' \mid q'^{\sigma'} \mid right'}
\end{array}
$$

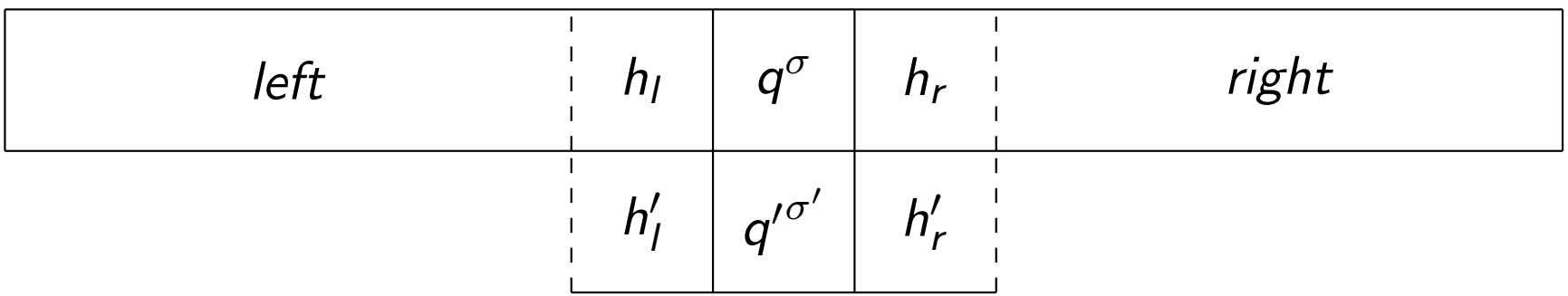
where $ls \sim_t^{z'} left, rs \sim_t^{z'} right$ and $ls' \sim_t^{z'} left', rs' \sim_t^{z'} right'$



Deterministic Simulation

$$q;(ls,\sigma,rs) \sim_c \boxed{\text{rev } left \mid q^{\sigma} \mid right}$$
$$\curlyvee \qquad\qquad \wr$$
$$q';(ls',\sigma',rs') \sim_c \boxed{\text{rev } left' \mid q'^{\sigma'} \mid right'}$$

where $ls \sim_t^{z'} left, rs \sim_t^{z'} right$ and $ls' \sim_t^{z'} left', rs' \sim_t^{z'} right'$



Deterministic Simulation

$$
\begin{array}{lll}
q;(ls,\sigma,rs) & \sim_c & \boxed{\text{rev } left \mid q^{\sigma} \mid right} \\
\Upsilon & & \wr \\
q';(ls',\sigma',rs') & \sim_c & \boxed{\text{rev } left' \mid q'^{\sigma'} \mid right'}
\end{array}
$$

where $ls \sim_t^{z'} left, rs \sim_t^{z'} right$ and $ls' \sim_t^{z'} left', rs' \sim_t^{z'} right'$

$left$	h_l	q^{σ}	h_r	$right$
$\exists! left'$	h'_l	$q'^{\sigma'}$	h'_r	$\exists! right'$

Tape Transformations

Add symbol:

$$rs \sim_t h \wedge |rs| < z' \rightarrow \exists! h', (h \rightsquigarrow \overrightarrow{a} :: h') \wedge a :: rs \sim_t^+ \overrightarrow{a} :: h'$$

$$ls \sim_t h \wedge |ls| < z' \rightarrow \exists! h', (\mathrm{rev} h \rightsquigarrow \mathrm{rev} \overleftarrow{a} :: h') \wedge a :: ls \sim_t^- \overleftarrow{a} :: h'$$

Remove symbol:

$$a :: b :: rs \sim_t a :: b :: h \rightarrow \exists! h', (a :: b :: h \rightsquigarrow \overleftarrow{b} :: h') \wedge b :: rs \sim_t^- \overleftarrow{b} :: h'$$

Leave unchanged:

$$a :: rs \sim_t a :: h \rightarrow \exists! h', (a :: h \rightsquigarrow \overline{a} :: h) \wedge a :: rs \sim_t^\circ \overline{a} :: h'$$

Main Simulation Results

Completeness

Let $(q, tape)$ be a configuration with $|tape| \leq k$. There exists s with $(q, tape) \sim_c s$.
If $(q, tape) \triangleright^{\leq t} (q', tape')$, then there exists s' with $s \rightsquigarrow^t s'$, $(q', tape') \sim_c s'$ and $s' \models R_{\text{final}}$.

Soundness

Let s be given such that $(q, tape) \sim_c s$ and $|tape| \leq k$ for some $q, tape$.
If $s \rightsquigarrow^t s'$ and $s' \models R_{\text{final}}$, then there exists $(q', tape')$ with $(q', tape') \sim_c s'$ such that $(q, tape) \triangleright^{\leq t} (q', tape')$ and $|tape'| \leq z'$.

Formulas: Representation of Predicates

$$f : \mathcal{F} := \mathsf{T} \mid v \mid f_1 \vee f_2 \mid f_1 \wedge f_2 \mid \neg f \qquad (v : \mathsf{var} := \mathbb{N})$$

- explicit assignments $e : \mathcal{L}(\mathbb{B})$ to variables $[s, s + |e|)$
- encoding of predicates $p : \mathcal{L}(\mathbb{B}) \to \mathbb{P}$:

  $\mathsf{encodesPredAt}\ \mathsf{start}\ l\ f\ p := \forall a, a \models f \leftrightarrow p\ (a[\mathsf{start}, \mathsf{start} + l])$

- $\mathsf{encodeBit}\ \mathsf{var}\ \mathsf{bit} :=$ **if** bit **then** var **else** $\neg$var

  $\mathsf{encodesPredAt}\ v\ 1\ (\mathsf{encodeBit}\ v\ b)(\lambda e.e = [s])$

The Tseytin Transformation

Goal: convert formula to CNF
Example: $f = f_1 \vee f_2$

Naive approach:

$$f = f_1 \vee f_2 \rightsquigarrow \quad N_1 \quad \vee \quad N_2$$

N : cnf $\quad$ C : clause $\quad$ v : var $\quad$ $f : \mathcal{F}$

The Tseytin Transformation

Goal: convert formula to CNF
Example: $f = f_1 \vee f_2$

Naive approach:

$$f = f_1 \vee f_2 \rightsquigarrow (C_1 \wedge C_2) \vee (C_3 \wedge C_4)$$

N : cnf $\quad$ C : clause $\quad$ v : var $\quad$ $f : \mathcal{F}$

The Tseytin Transformation

Goal: convert formula to CNF
Example: $f = f_1 \vee f_2$

Naive approach:

$$f = f_1 \vee f_2 \rightsquigarrow (C_1 \wedge C_2) \vee (C_3 \wedge C_4)$$
$$\leftrightarrow (C_1 \vee C_3) \wedge (C_1 \vee C_4) \wedge (C_2 \vee C_3) \wedge (C_2 \wedge C_4)$$

N : cnf $\quad$ C : clause $\quad$ v : var $\quad$ $f : \mathcal{F}$

The Tseytin Transformation

Goal: convert formula to CNF
Example: $f = f_1 \vee f_2$

Naive approach:

$$f = f_1 \vee f_2 \rightsquigarrow (C_1 \wedge C_2) \vee (C_3 \wedge C_4)$$
$$\leftrightarrow (C_1 \vee C_3) \wedge (C_1 \vee C_4) \wedge (C_2 \vee C_3) \wedge (C_2 \wedge C_4)$$

exponential blowup!

N : cnf $\quad$ C : clause $\quad$ v : var $\quad$ $f : \mathcal{F}$

The Tseytin Transformation

Goal: convert formula to CNF
Example: $f = f_1 \vee f_2$

Solution: introduce new variables: $f \mapsto (v, N)$ s.t.

$$f \left|\!\!\models\!\!\right| (v, N) := \mathbf{SAT} f \leftrightarrow \mathbf{SAT}(v \wedge N)$$

N : cnf $\quad$ C : clause $\quad$ v : var $\quad$ $f : \mathcal{F}$

The Tseytin Transformation

Goal: convert formula to CNF
Example: $f = f_1 \vee f_2$

Solution: introduce new variables: $f \mapsto (v, N)$ s.t.

$$f \left|\!\models\!\right| (v, N) := \mathbf{SAT} f \leftrightarrow \mathbf{SAT}(v \wedge N)$$

$$f_1 \left|\!\models\!\right| (v_1, N_1) \qquad f_2 \left|\!\models\!\right| (v_2, N_2)$$

Goal: $f = f_1 \vee f_2 \left|\!\models\!\right| (v, N)$ for fresh v

N : cnf $\qquad$ C : clause $\qquad$ v : var $\qquad$ $f : \mathcal{F}$

The Tseytin Transformation

Goal: convert formula to CNF
Example: $f = f_1 \vee f_2$

Solution: introduce new variables: $f \mapsto (v, N)$ s.t.

$$f \models\!\!\!\models (v, N) := \mathbf{SAT} f \leftrightarrow \mathbf{SAT}(v \wedge N)$$

$$f_1 \models\!\!\!\models (v_1, N_1) \qquad f_2 \models\!\!\!\models (v_2, N_2)$$

Goal: $f = f_1 \vee f_2 \models\!\!\!\models (v, N)$ for fresh v

$$N := \underbrace{N_1}_{f_1 \leftrightarrow v_1} \wedge \underbrace{N_2}_{f_2 \leftrightarrow v_2} \wedge ?$$

N : cnf $\qquad$ C : clause $\qquad$ v : var $\qquad$ $f : \mathcal{F}$

The Tseytin Transformation

Goal: convert formula to CNF
Example: $f = f_1 \vee f_2$

Solution: introduce new variables: $f \mapsto (v, N)$ s.t.

$$f \models\!\!\models (v, N) := \mathbf{SAT} f \leftrightarrow \mathbf{SAT}(v \wedge N)$$

$$f_1 \models\!\!\models (v_1, N_1) \qquad f_2 \models\!\!\models (v_2, N_2)$$

Goal: $f = f_1 \vee f_2 \models\!\!\models (v, N)$ for fresh v

$$N := \underbrace{N_1}_{f_1 \leftrightarrow v_1} \wedge \underbrace{N_2}_{f_2 \leftrightarrow v_2} \wedge (v \leftrightarrow (v_1 \vee v_2))$$

$$\Rightarrow f \leftrightarrow v$$

N : cnf $\qquad$ C : clause $\qquad$ v : var $\qquad$ $f : \mathcal{F}$

Tseytin: Correctness Relation

$$\text{tseytin}' : \underbrace{\text{var}}_{\text{next free var}} \to \mathcal{F} \to \underbrace{\text{var}}_{\text{representing variable}} \times \text{cnf} \times \underbrace{\text{var}}_{\text{next free var}}$$

$$f \left|\!\!\left|\!\!\frac{nf, nf'}{b}\right|\!\!\right| (v, N), \text{ iff}$$

- $N \subseteq ([0, b) \cup [nf, nf'))$,
- $v \in [nf, nf')$,
- for all $a \subseteq [0, b)$, there exists $a' \subseteq [nf, nf')$ such that $(a \cup a') \models N$,
- and for all a with $a \models N$, the equivalence $a \models v \leftrightarrow a \models f$ holds.

Basic Definitions: NP[2]

Problem $Q : X \to \mathbb{P}$

$Q \in \mathrm{NP}$ iff:

- there is a verifier $V : X \to Y \to \mathbb{P}$
- V is polynomial-time computable
- V verifies Q
  - $V\ x\ y \to Q\ x$
  - $Q\ x \to \exists y, V\ x\ y$ and y has polynomial size

[2] Definitions by Fabian

Basic Definitions: Polynomial-Time Reductions[3]

$$P : X \to \mathbb{P} \qquad Q : Y \to \mathbb{P}$$

$$P \preceq_p Q := \exists f : X \to Y, \quad (\forall x, P\ x \leftrightarrow Q(f\ x))$$
$$\wedge \text{ polynomial-time computable } f$$

[3] Definitions by Fabian

Basic Definitions: Polynomial-Time Reductions[3]

$$P : X \to \mathbb{P} \qquad Q : Y \to \mathbb{P}$$

$$P \preceq_p Q := \exists f : X \to Y, \quad (\forall x, P\ x \leftrightarrow Q(f\ x))$$
$$\wedge \text{ polynomial-time computable } f$$
$$\wedge\ f \text{ increases the size polynomially}$$

[3] Definitions by Fabian

Basic Definitions: Polynomial-Time Reductions[3]

$$P : X \to \mathbb{P} \qquad Q : Y \to \mathbb{P}$$

$$P \preceq_p Q := \exists f : X \to Y, \quad (\forall x, P\ x \leftrightarrow Q(f\ x))$$
$$\wedge \text{ polynomial-time computable } f$$
$$\wedge\ f \text{ increases the size polynomially}$$

$$\text{NP-hard } P := \forall Q, Q \in \text{NP} \to Q \preceq_p P$$
$$\text{NP-complete } P := P \in \text{NP} \wedge \text{NP-hard } P$$

[3] Definitions by Fabian

Proving Time Bounds

$$\begin{aligned}
\text{cnf} &:= \mathcal{L}(\text{clause}) \\
\mathcal{E}_N &: \text{assgn} \to \text{cnf} \to \mathbb{B} \\
\mathcal{E}_N\ a\ [\,] &:= \mathsf{T} \\
\mathcal{E}_N\ a\ (C :: N) &:= \mathcal{E}_C\ a\ C\ \&\ \mathcal{E}_N\ a\ N
\end{aligned}$$

1. extraction, running time in terms of arguments

$$\begin{aligned}
T(\mathcal{E}_N)\ a\ [\,] &\geq 9 \\
T(\mathcal{E}_N)\ a\ (C :: N) &\geq T(\mathcal{E}_N)\ a\ N + T(\mathcal{E}_C)\ a\ C + 22
\end{aligned}$$

2. bound by monotonic polynomial $p_{\mathcal{E}_N} : \mathbb{N} \to \mathbb{N}$ s.t.

$$\forall a\ N, T_{\mathcal{E}_N}\ a\ N \leq p_{\mathcal{E}_N}(\|\overline{a}\| + \left\|\overline{N}\right\|)$$

in this case: $p_{\mathcal{E}_N}(n) := n \cdot p_{\mathcal{E}_C}(n) + c_{\mathcal{E}_N} \cdot (n+1)$

#P

counting problem $f : X \to \mathbb{N}$

Verifier Characterisation of #P

$f \in \#\mathrm{P}$, iff there exists a verifier $R : X \to Y \to \mathbb{P}$ s.t.

- R runs in polynomial time
- R only accepts certificates of a polynomial size
- $\forall x, |\{y \mid R\ x\ y\}| = f\ x$

finite cardinalities $|\{y \mid R\ x\ y\}|$ defined using listability

Shortcomings:

- only computable counting problems definable (solution: functional relations)
- precise definition of counting problems requires a bit of care (e.g. **#SAT**)

References


Asperti, A. and Ricciotti, W. (2015).
A formalization of multi-tape turing machines.
Theoretical Computer Science, 603.


Bläser, M.
Theoretical computer science: An introduction.


Cook, S. A. (1971).
The complexity of theorem-proving procedures.
In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA. ACM.

References


Forster, Y. and Kunze, F. (2019).
A certifying extraction with time bounds from coq to call-by-value lambda-calculus.
In *Interactive Theorem Proving - 10th International Conference, ITP 2019, Portland, USA*.
Also available as arXiv:1904.11818.


Forster, Y., Kunze, F., and Roth, M. (2019).
The weak call-by-value lambda-calculus is reasonable for both time and space.
Technical report.
Full version appeared as arXiv:1902.07515 To appear.

References


Forster, Y., Kunze, F., and Wuttke, M. (2020).
Verified programming of turing machines in coq.
In *9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20–21, 2020*, New York, NY, USA. ACM.


Gamboa, R. and Cowles, J. (2004).
A mechanical proof of the cook-levin theorem.
In Slind, K., Bunker, A., and Gopalakrishnan, G., editors, *Theorem Proving in Higher Order Logics*, pages 99–116, Berlin, Heidelberg. Springer Berlin Heidelberg.


Karp, R. M. (1972).
Reducibility among Combinatorial Problems, pages 85–103.
Springer US, Boston, MA.

References


Levin, L. A. (1973).
Universal sequential search problems.
volume 9, pages 265 – 266.
[Probl. Peredachi Inf., **9**,3:115-116, 1973].


Sipser, M. (1997).
Introduction to Theory of Computation.
PWS Publishing Company, 1 edition.