Lecture 6: Cook Levin Theorem

Arijit Bishnu

11.03.2010

Outline

Outline

Warm Up

What is needed to be proved?

- We need to find a poly-time reduction that turns any $x \in \{0,1\}^*$ into a CNF formula φ_x such that $x \in L$ iff φ_x is satisfiable.

Warm Up

What is needed to be proved?

- We need to find a poly-time reduction that turns any $x \in \{0,1\}^*$ into a CNF formula φ_x such that $x \in L$ iff φ_x is satisfiable.
- We only know that $L \in$ NP. The reduction has to rely only on the definition of computation.

Warm Up

What is needed to be proved?

- We need to find a poly-time reduction that turns any $x \in \{0,1\}^*$ into a CNF formula φ_x such that $x \in L$ iff φ_x is satisfiable.
- We only know that $L \in$ NP. The reduction has to rely only on the definition of computation.
- We take help of the fact that any algorithm that takes a fixed number $|x|$ of bits as input and produces a yes/no answer can be represented by a circuit equivalent to a CNF formula.

Warm Up

What is needed to be proved?

- We need to find a poly-time reduction that turns any $x \in \{0,1\}^*$ into a CNF formula φ_x such that $x \in L$ iff φ_x is satisfiable.
- We only know that $L \in$ NP. The reduction has to rely only on the definition of computation.
- We take help of the fact that any algorithm that takes a fixed number $|x|$ of bits as input and produces a yes/no answer can be represented by a circuit equivalent to a CNF formula.
- The circuit has to be equivalent to the algorithm, i.e. its output is 1 precisely on those inputs for which the algorithm outputs yes.

Warm Up

What is needed to be proved?

- We need to find a poly-time reduction that turns any $x \in \{0,1\}^*$ into a CNF formula φ_x such that $x \in L$ iff φ_x is satisfiable.
- We only know that $L \in$ NP. The reduction has to rely only on the definition of computation.
- We take help of the fact that any algorithm that takes a fixed number $|x|$ of bits as input and produces a yes/no answer can be represented by a circuit equivalent to a CNF formula.
- The circuit has to be equivalent to the algorithm, i.e. its output is 1 precisely on those inputs for which the algorithm outputs yes.
- If the algorithm takes number of steps that is polynomial in $|x|$, the circuit will also be of size polynomial in $|x|$.

Warm Up

What is needed to be proved?

- We are trying to show that $L \leq_P$ SAT.

Warm Up

What is needed to be proved?

- We are trying to show that $L \leq_P$ SAT.
- So, given an input x, we want to decide whether $x \in L$ using a black box that can solve instances of SAT.

Warm Up

What is needed to be proved?

- We are trying to show that $L \leq_P$ SAT.
- So, given an input x, we want to decide whether $x \in L$ using a black box that can solve instances of SAT.
- We know that $L \in$ NP, i.e. L has an efficient certifier $M(\cdot\ ,\ \cdot)$.

Warm Up

What is needed to be proved?

- We are trying to show that $L \leq_P$ SAT.
- So, given an input x, we want to decide whether $x \in L$ using a black box that can solve instances of SAT.
- We know that $L \in$ NP, i.e. L has an efficient certifier $M(\cdot\ ,\ \cdot)$.
- So, to determine whether $x \in L$, for some specific input of length $|x|$, we need to answer: Is there a u, $|u| = p(|x|)$, such that $M(x, u) = 1$?

Warm Up

Proof Idea

- We need the answer only for a specific input x.

Warm Up

Proof Idea

- We need the answer only for a specific input x.
- We view $M(\cdot\ ,\ \cdot)$ as an algorithm on $|x| + |u|$ bits.

Warm Up

Proof Idea

- We need the answer only for a specific input x.
- We view $M(\cdot\ ,\ \cdot)$ as an algorithm on $|x| + |u|$ bits.
- Convert M to a poly-size circuit K with $|x| + |u|$ sources.

Warm Up

Proof Idea

- We need the answer only for a specific input x.
- We view $M(\cdot\ ,\ \cdot)$ as an algorithm on $|x| + |u|$ bits.
- Convert M to a poly-size circuit K with $|x| + |u|$ sources.
- The first $|x|$ sources will be hard-coded with the values of the bits in x.

Warm Up

Proof Idea

- We need the answer only for a specific input x.
- We view $M(\cdot\ ,\ \cdot)$ as an algorithm on $|x| + |u|$ bits.
- Convert M to a poly-size circuit K with $|x| + |u|$ sources.
- The first $|x|$ sources will be hard-coded with the values of the bits in x.
- The remaining $|u|$ sources will be labeled with variables representing the bits of u; these will be inputs to the circuit K.

Warm Up

Proof Idea

- We need the answer only for a specific input x.
- We view $M(\cdot\ ,\ \cdot)$ as an algorithm on $|x| + |u|$ bits.
- Convert M to a poly-size circuit K with $|x| + |u|$ sources.
- The first $|x|$ sources will be hard-coded with the values of the bits in x.
- The remaining $|u|$ sources will be labeled with variables representing the bits of u; these will be inputs to the circuit K.
- Observe that $x \in L$ iff there is a way to set the input bits to K so that K produces an output of 1.

Outline

Expressiveness of Boolean Formula

Claim

For every boolean function $f : \{0,1\}^\ell \to \{0,1\}$ there is an ℓ-variable CNF formula φ of size $\ell 2^\ell$ s.t. $\varphi(u) = f(u)$ for every $u \in \{0,1\}^\ell$, where the size of a CNF formula is defined to be the number of $\vee/\wedge$ symbols it contains.

The Essence

A CNF formulae of sufficient size can express every Boolean condition.

Proof of Claim

Proof

- For every $v \in \{0,1\}^{\ell}$, $\exists$ a clause C_v s.t. $C_v(v) = 0$ and $C_v(u) = 1$ for every $u \neq v$. For example, if $v =< 1,0,1,0 >$, then the corr. clause is $\overline{u_1} \vee u_2 \vee \overline{u_3} \vee u_4$.

Proof of Claim

Proof

- For every $v \in \{0,1\}^{\ell}$, $\exists$ a clause C_v s.t. $C_v(v) = 0$ and $C_v(u) = 1$ for every $u \neq v$. For example, if $v =< 1,0,1,0 >$, then the corr. clause is $\overline{u_1} \vee u_2 \vee \overline{u_3} \vee u_4$.
- Let $\varphi = \bigvee C_v$ for v s.t. $f(v) = 0$. $|\varphi| = \ell 2^{\ell}$

Proof of Claim

Proof

- For every $v \in \{0,1\}^{\ell}$, $\exists$ a clause C_v s.t. $C_v(v) = 0$ and $C_v(u) = 1$ for every $u \neq v$. For example, if $v =< 1,0,1,0 >$, then the corr. clause is $\overline{u_1} \vee u_2 \vee \overline{u_3} \vee u_4$.
- Let $\varphi = \bigvee C_v$ for v s.t. $f(v) = 0$. $|\varphi| = \ell 2^{\ell}$
- Then for every u s.t. $f(u) = 0$ it holds that $C_u(u) = 0$ and hence, $\varphi(u) = 0$.

Proof of Claim

Proof

- For every $v \in \{0,1\}^{\ell}$, $\exists$ a clause C_v s.t. $C_v(v) = 0$ and $C_v(u) = 1$ for every $u \neq v$. For example, if $v =< 1,0,1,0 >$, then the corr. clause is $\overline{u_1} \vee u_2 \vee \overline{u_3} \vee u_4$.
- Let $\varphi = \bigvee C_v$ for v s.t. $f(v) = 0$. $|\varphi| = \ell 2^{\ell}$
- Then for every u s.t. $f(u) = 0$ it holds that $C_u(u) = 0$ and hence, $\varphi(u) = 0$.
- On the other hand, if $f(u) = 1$ then $C_v(u) = 1$ for every v s.t. $f(v) = 0$ and hence, $\varphi(u) = 1$.

Proof of Claim

Proof

- For every $v \in \{0,1\}^\ell$, $\exists$ a clause C_v s.t. $C_v(v) = 0$ and $C_v(u) = 1$ for every $u \neq v$. For example, if $v =< 1,0,1,0 >$, then the corr. clause is $\overline{u_1} \vee u_2 \vee \overline{u_3} \vee u_4$.
- Let $\varphi = \bigvee C_v$ for v s.t. $f(v) = 0$. $|\varphi| = \ell 2^\ell$
- Then for every u s.t. $f(u) = 0$ it holds that $C_u(u) = 0$ and hence, $\varphi(u) = 0$.
- On the other hand, if $f(u) = 1$ then $C_v(u) = 1$ for every v s.t. $f(v) = 0$ and hence, $\varphi(u) = 1$.
- So, we get that for every u, $\varphi(u) = f(u)$.

Outline

SAT is NP-complete

SAT is in NP

Easy to show.

SAT is NP-complete

SAT is in NP

Easy to show.

Each language $A \in$ NP is poly-time reducible to SAT

SAT is NP-complete

SAT is in NP

Easy to show.

Each language $A \in$ NP is poly-time reducible to SAT

- The reduction for A takes a string w and produces a Boolean formula φ that simulates the working of the NDTM for A on w.

SAT is NP-complete

SAT is in NP

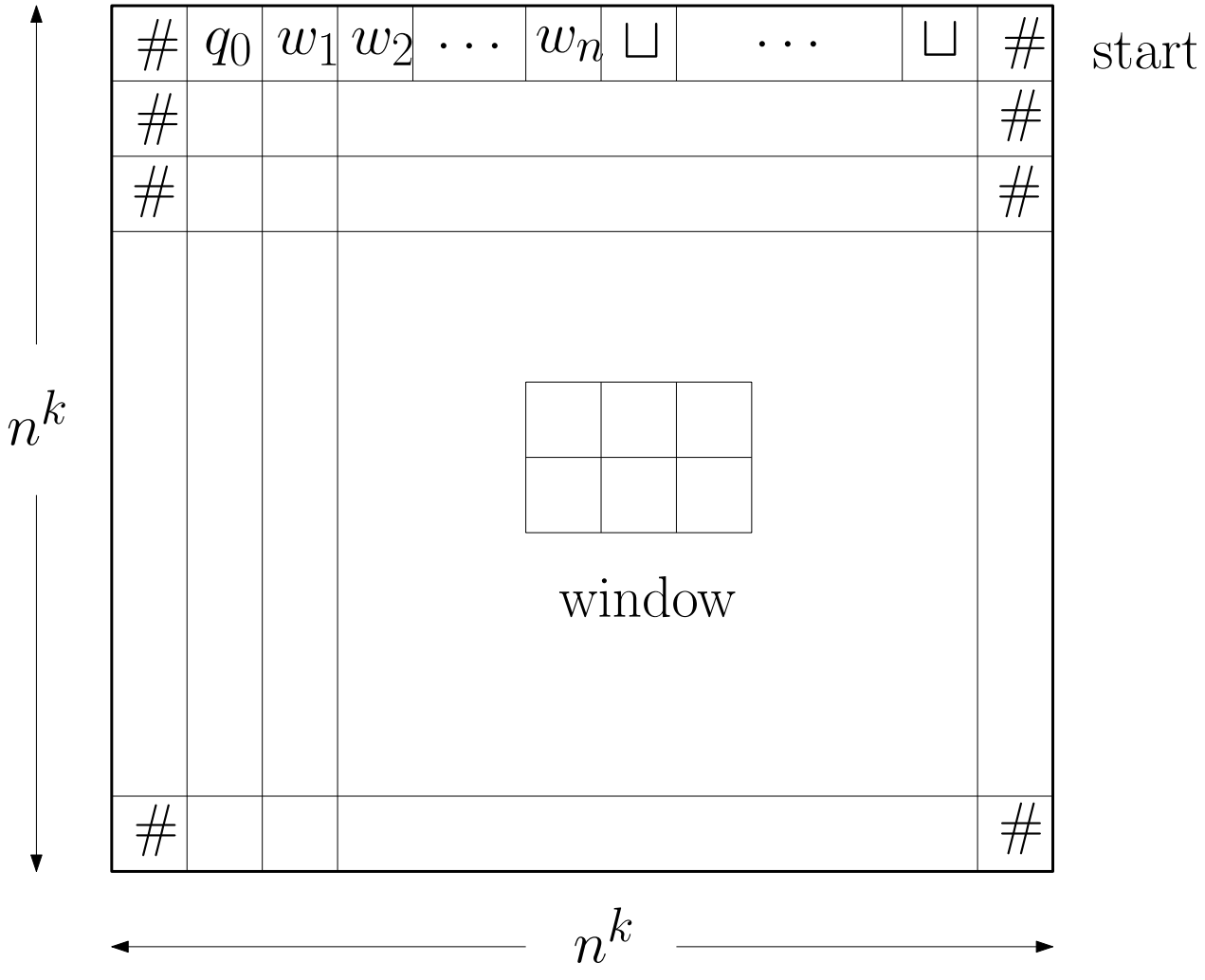
Easy to show.

Each language $A \in$ NP is poly-time reducible to SAT

- The reduction for A takes a string w and produces a Boolean formula φ that simulates the working of the NDTM for A on w.
- The idea is to design a φ such that $w \in A$ iff φ is satisfiable.

A Proof of Cook Levin using Tableau

Let N be a NDTM that decides A in n^k time for some constant k.



A tableau is an $n^k \times n^k$ table of configurations.

Understanding the Tableau

- A tableau for N on w ($|w| = n$) is an $n^k \times n^k$ table whose rows are the configurations of a branch of the computation of N on w.

Understanding the Tableau

- A tableau for N on w ($|w| = n$) is an $n^k \times n^k$ table whose rows are the configurations of a branch of the computation of N on w.
- Each row (configuration) starts and ends with a #.

Understanding the Tableau

- A tableau for N on w ($|w| = n$) is an $n^k \times n^k$ table whose rows are the configurations of a branch of the computation of N on w.
- Each row (configuration) starts and ends with a #.
- The first row is the start configuration of N on w and each row follows the previous one according to N's transition function.

Understanding the Tableau

- A tableau for N on w ($|w| = n$) is an $n^k \times n^k$ table whose rows are the configurations of a branch of the computation of N on w.
- Each row (configuration) starts and ends with a #.
- The first row is the start configuration of N on w and each row follows the previous one according to N's transition function.
- A tableau is accepting if any row of the tableau is an accepting configuration.

Understanding the Tableau

- A tableau for N on w ($|w| = n$) is an $n^k \times n^k$ table whose rows are the configurations of a branch of the computation of N on w.
- Each row (configuration) starts and ends with a $\#$.
- The first row is the start configuration of N on w and each row follows the previous one according to N's transition function.
- A tableau is accepting if any row of the tableau is an accepting configuration.
- The problem of whether N accepts $w \Leftrightarrow$ the problem of determining whether an accepting tableau for N on w exists.

The Poly-time Reduction: On input w, produce a φ

Fix the variables of φ

The Poly-time Reduction: On input w, produce a φ

Fix the variables of φ

- Let Q be the state set and Γ be the tape alphabet set of N.

The Poly-time Reduction: On input w, produce a φ

Fix the variables of φ

- Let Q be the state set and Γ be the tape alphabet set of N.
- Let $C = Q \cup \Gamma \cup \{\#\}$.

The Poly-time Reduction: On input w, produce a φ

Fix the variables of φ

- Let Q be the state set and Γ be the tape alphabet set of N.
- Let $C = Q \cup \Gamma \cup \{\#\}$.
- For each $1 \leq i, j \leq n^k$ and for each $s \in C$, we have a variable $x_{i,j,s}$. (How many variables?)

The Poly-time Reduction: On input w, produce a φ

Fix the variables of φ

- Let Q be the state set and Γ be the tape alphabet set of N.
- Let $C = Q \cup \Gamma \cup \{\#\}$.
- For each $1 \leq i, j \leq n^k$ and for each $s \in C$, we have a variable $x_{i,j,s}$. (How many variables?)
- Each of the $(n^k)^2$ entries of the tableau is called a cell. $cell[i, j]$ contains a symbol from C.

The Poly-time Reduction: On input w, produce a φ

Fix the variables of φ

- Let Q be the state set and Γ be the tape alphabet set of N.
- Let $C = Q \cup \Gamma \cup \{\#\}$.
- For each $1 \leq i, j \leq n^k$ and for each $s \in C$, we have a variable $x_{i,j,s}$. (How many variables?)
- Each of the $(n^k)^2$ entries of the tableau is called a cell. $cell[i, j]$ contains a symbol from C.
- We represent the contents of the cells with the variables of φ. $x_{i,j,s} = 1$ implies $cell[i, j]$ contains an s.

The Poly-time Reduction: On input w, produce a φ

Fix the variables of φ

- Let Q be the state set and Γ be the tape alphabet set of N.
- Let $C = Q \cup \Gamma \cup \{\#\}$.
- For each $1 \leq i, j \leq n^k$ and for each $s \in C$, we have a variable $x_{i,j,s}$. (How many variables?)
- Each of the $(n^k)^2$ entries of the tableau is called a cell. $cell[i, j]$ contains a symbol from C.
- We represent the contents of the cells with the variables of φ. $x_{i,j,s} = 1$ implies $cell[i, j]$ contains an s.
- We design φ so that a satisfying assignment to the variables corresponds to an accepting tableau for N on w.

The Poly-time Reduction: On input w, produce a φ

Fix the variables of φ

- Let Q be the state set and Γ be the tape alphabet set of N.
- Let $C = Q \cup \Gamma \cup \{\#\}$.
- For each $1 \leq i, j \leq n^k$ and for each $s \in C$, we have a variable $x_{i,j,s}$. (How many variables?)
- Each of the $(n^k)^2$ entries of the tableau is called a cell. $cell[i, j]$ contains a symbol from C.
- We represent the contents of the cells with the variables of φ. $x_{i,j,s} = 1$ implies $cell[i, j]$ contains an s.
- We design φ so that a satisfying assignment to the variables corresponds to an accepting tableau for N on w.
- To describe φ, we need to encode the variables, start configuration, moves of N and accepting state.

The Poly-time Reduction: On input w, produce a φ

Fix the variables of φ

- Let Q be the state set and Γ be the tape alphabet set of N.
- Let $C = Q \cup \Gamma \cup \{\#\}$.
- For each $1 \leq i, j \leq n^k$ and for each $s \in C$, we have a variable $x_{i,j,s}$. (How many variables?)
- Each of the $(n^k)^2$ entries of the tableau is called a cell. $cell[i, j]$ contains a symbol from C.
- We represent the contents of the cells with the variables of φ. $x_{i,j,s} = 1$ implies $cell[i, j]$ contains an s.
- We design φ so that a satisfying assignment to the variables corresponds to an accepting tableau for N on w.
- To describe φ, we need to encode the variables, start configuration, moves of N and accepting state.
- Thus, $\varphi = \varphi_{\text{cell}} \wedge \varphi_{\text{start}} \wedge \varphi_{\text{move}} \wedge \varphi_{\text{accept}}$..

Formula for φ_{cell}

- Setting $x_{i,j,s} = 1$ corresponds to placing symbols s in $cell[i, j]$.

Formula for φ_{cell}

- Setting $x_{i,j,s} = 1$ corresponds to placing symbols s in $cell[i,j]$.
- To obtain a correspondence between an assignment and a tableau, we must ensure that the assignment sets to 1 exactly one variable for each cell. The following formula does it.

$$\varphi_{\text{cell}} = \bigwedge_{1 \le i,j \le n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

Formula for φ_{cell}

- Setting $x_{i,j,s} = 1$ corresponds to placing symbols s in $cell[i,j]$.
- To obtain a correspondence between an assignment and a tableau, we must ensure that the assignment sets to 1 exactly one variable for each cell. The following formula does it.

$$\varphi_{\text{cell}} = \bigwedge_{1 \le i,j \le n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

- As an example, if $C = \{s_1, s_2, \ldots, s_l\}$, then $\bigvee_{s \in C} x_{i,j,s}$ means $x_{i,j,s_1} \vee x_{i,j,s_2} \vee \ldots \vee x_{i,j,s_l}$.

Formula for φ_{cell}

- Setting $x_{i,j,s} = 1$ corresponds to placing symbols s in $cell[i,j]$.
- To obtain a correspondence between an assignment and a tableau, we must ensure that the assignment sets to 1 exactly one variable for each cell. The following formula does it.

$$\varphi_{\text{cell}} = \bigwedge_{1 \le i,j \le n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

- As an example, if $C = \{s_1, s_2, \ldots, s_l\}$, then $\bigvee_{s \in C} x_{i,j,s}$ means $x_{i,j,s_1} \vee x_{i,j,s_2} \vee \ldots \vee x_{i,j,s_l}$.
- $\bigvee_{s \in C} x_{i,j,s}$ says that at least one variable that is associated to each cell is 1.

Formula for φ_{cell}

- Setting $x_{i,j,s} = 1$ corresponds to placing symbols s in $cell[i,j]$.
- To obtain a correspondence between an assignment and a tableau, we must ensure that the assignment sets to 1 exactly one variable for each cell. The following formula does it.

$$\varphi_{\text{cell}} = \bigwedge_{1 \le i,j \le n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

- As an example, if $C = \{s_1, s_2, \ldots, s_l\}$, then $\bigvee_{s \in C} x_{i,j,s}$ means $x_{i,j,s_1} \vee x_{i,j,s_2} \vee \ldots \vee x_{i,j,s_l}$.
- $\bigvee_{s \in C} x_{i,j,s}$ says that at least one variable that is associated to each cell is 1.
- $\bigwedge_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}})$ says that no more than one variable is turned 1.

Formula for φ_{cell}

- Setting $x_{i,j,s} = 1$ corresponds to placing symbols s in $cell[i,j]$.
- To obtain a correspondence between an assignment and a tableau, we must ensure that the assignment sets to 1 exactly one variable for each cell. The following formula does it.

$$\varphi_{\text{cell}} = \bigwedge_{1 \le i,j \le n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

- As an example, if $C = \{s_1, s_2, \ldots, s_l\}$, then $\bigvee_{s \in C} x_{i,j,s}$ means $x_{i,j,s_1} \vee x_{i,j,s_2} \vee \ldots \vee x_{i,j,s_l}$.
- $\bigvee_{s \in C} x_{i,j,s}$ says that at least one variable that is associated to each cell is 1.
- $\bigwedge_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}})$ says that no more than one variable is turned 1.
- Thus, any satisfying assignment specifies one symbol in each cell of the tableau.

Formula for φ_{start}

We need to take care of φ_{start}, φ_{move} and φ_{accept} to ensure that we have an accepting tableau.

Formula for φ_{start}

We need to take care of φ_{start}, φ_{move} and φ_{accept} to ensure that we have an accepting tableau.

- φ_{start} ensures that the first row of the table is the starting configuration of N on w by explicitly making the corresponding variables 1.

Formula for φ_{start}

We need to take care of φ_{start}, φ_{move} and φ_{accept} to ensure that we have an accepting tableau.

- φ_{start} ensures that the first row of the table is the starting configuration of N on w by explicitly making the corresponding variables 1.
- Thus, we have

$$\begin{aligned} \varphi_{\text{start}} = \; & x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ & x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \ldots \wedge x_{1,n+2,w_n} \wedge \\ & x_{1,n+3,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#} \end{aligned} \tag{1}$$

Formula for φ_{accept}

- Formula φ_{accept} guarantees that an accepting configuration occurs in the tableau.

Formula for φ_{accept}

- Formula φ_{accept} guarantees that an accepting configuration occurs in the tableau.
- The formula ensures that q_{accept}, the symbol for the accept state, occurs in one of the cells of the tableau by stipulating that one of the corresponding variables is set to 1.

Formula for φ_{accept}

- Formula φ_{accept} guarantees that an accepting configuration occurs in the tableau.
- The formula ensures that q_{accept}, the symbol for the accept state, occurs in one of the cells of the tableau by stipulating that one of the corresponding variables is set to 1.
- Thus, the formula is

$$\varphi_{\text{accept}} = \bigvee_{1 \leq i,j \leq n^k} x_{i,j,q_{\text{accept}}}$$

Formula for φmove

- This formula guarantees that each row of the table corresponds to a configuration that legally follows the preceding row's configuration according to δ, the transition function of N.

Formula for φmove

- This formula guarantees that each row of the table corresponds to a configuration that legally follows the preceding row's configuration according to δ, the transition function of N.
- The formula does so by ensuring that each 2×3 window of cells is legal. (Why a 2×3 window and not a 3×5 window?)

Formula for φmove

- This formula guarantees that each row of the table corresponds to a configuration that legally follows the preceding row's configuration according to δ, the transition function of N.
- The formula does so by ensuring that each 2×3 window of cells is legal. (Why a 2×3 window and not a 3×5 window?)
- A 2×3 window is legal if that window does not violate the actions specified by δ.

Examples of Legal Windows

$\Gamma = \{a, b, c\}$ and $Q = \{q_1, q_2\}$. Assume that when in state q_1, with the head reading an a, N writes a b, stays in state q_1 and moves right; and that when in state q_1 with the head reading a b, N nondeterministically either

1. writes a c, enters q_2 and moves to the left, OR
2. writes an a, enters q_2 and moves to the right

a	q_1	b
q_2	a	c

a	q_1	b
a	a	q_2

a	a	q_1
a	a	b

#	b	a
#	b	a

a	b	a
a	b	q_2

b	b	b
c	b	b

Examples of Illegal Windows

Solve them as an Exercise to find why they are Illegal?

a	b	a
a	a	a

a	q_1	b
q_1	a	a

b	q_1	b
q_2	b	q_2

A Claim

Claim

If the top row of the table is the start configuration and every window in the table is legal, each row of the table is a configuration that legally follows the preceding one.

Proof

- Consider any two adjacent configurations (rows) in the tableau - the upper and lower configurations.

A Claim

Claim

If the top row of the table is the start configuration and every window in the table is legal, each row of the table is a configuration that legally follows the preceding one.

Proof

- Consider any two adjacent configurations (rows) in the tableau - the upper and lower configurations.
- In the upper configuration, every cell that is not adjacent to a state symbol and that does not contain #, is the centre top cell whose top row contains no states. Therefore, the symbol must appear unchanged in the centre bottom of the window.

A Claim

Claim

If the top row of the table is the start configuration and every window in the table is legal, each row of the table is a configuration that legally follows the preceding one.

Proof

- Consider any two adjacent configurations (rows) in the tableau - the upper and lower configurations.
- In the upper configuration, every cell that is not adjacent to a state symbol and that does not contain #, is the centre top cell whose top row contains no states. Therefore, the symbol must appear unchanged in the centre bottom of the window.
- The window containing the state symbol in the centre top cell guarantees that the corresponding three positions are updated according to δ of N. So, if the upper configuration is legal, so is the lower one as it follows δ of N.

Finally the Formula for φ_{move}

- The construction of φ_{move} should ensure that all the windows in the tableau are legal.

Finally the Formula for φ_{move}

- The construction of φ_{move} should ensure that all the windows in the tableau are legal.
- Each window has six cells which may be set in a fixed number of ways to yield a legal window. So, the overall picture of the expression is:

$$\varphi_{\text{move}} = \bigwedge_{1<i\leq n^k, 1<j<n^k} (\textit{the}(i,j) \text{ window is legal}).$$

Finally the Formula for φmove

- The construction of φ_{move} should ensure that all the windows in the tableau are legal.
- Each window has six cells which may be set in a fixed number of ways to yield a legal window. So, the overall picture of the expression is:

$$\varphi\text{move} = \bigwedge_{1<i\leq n^k, 1<j<n^k} (\textit{the}(i,j) \text{ window is legal}).$$

- To make the term ($\textit{the}(i,j)$ window is legal) more concrete, we write the contents of the six cells as $a_1, \ldots, a_6$. The term is as follows:

$$\bigvee_{\mathcal{R}} (x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a_6}).$$

$$\text{Predicate } \mathcal{R} = \{a_1, \ldots, a_6 \text{ is a legal window } \}$$

The Complexity of the Reduction

Is it polynomial in $n = |w|$?

- The tableau has $n^k \times n^k = n^{2k}$ cells.

The Complexity of the Reduction

Is it polynomial in $n = |w|$?

- The tableau has $n^k \times n^k = n^{2k}$ cells.
- Recall that $C = Q \cup \Gamma \cup \{\#\}$ and notice that C does not depend on $|w| = n$ but on the NDTM N. Each cell has $|C| = I$ variables. As I does not depend on n, the total number of variables is $O(n^{2k})$.

The Complexity of the Reduction

Is it polynomial in $n = |w|$?

- The tableau has $n^k \times n^k = n^{2k}$ cells.
- Recall that $C = Q \cup \Gamma \cup \{\#\}$ and notice that C does not depend on $|w| = n$ but on the NDTM N. Each cell has $|C| = l$ variables. As l does not depend on n, the total number of variables is $O(n^{2k})$.
- Now, focus on the formulae for each of φ_{cell}, φ_{start}, φ_{move} and φ_{accept}.

The Complexity of the Reduction

Is it polynomial in $n = |w|$?

- The tableau has $n^k \times n^k = n^{2k}$ cells.
- Recall that $C = Q \cup \Gamma \cup \{\#\}$ and notice that C does not depend on $|w| = n$ but on the NDTM N. Each cell has $|C| = l$ variables. As l does not depend on n, the total number of variables is $O(n^{2k})$.
- Now, focus on the formulae for each of φ_{cell}, φ_{start}, φ_{move} and φ_{accept}.
- φ_{start} has a fragment for each cell in the top row. So, it is of size $O(n^k)$.

The Complexity of the Reduction

Is it polynomial in $n = |w|$?

- The tableau has $n^k \times n^k = n^{2k}$ cells.
- Recall that $C = Q \cup \Gamma \cup \{\#\}$ and notice that C does not depend on $|w| = n$ but on the NDTM N. Each cell has $|C| = l$ variables. As l does not depend on n, the total number of variables is $O(n^{2k})$.
- Now, focus on the formulae for each of φ_{cell}, φ_{start}, φ_{move} and φ_{accept}.
- φ_{start} has a fragment for each cell in the top row. So, it is of size $O(n^k)$.
- φ_{cell}, φ_{move} and φ_{accept} each contain a fixed-size fragment of the formula for each cell of the tableau. So, they are of size $O(n^{2k})$. So, φ is polynomial in size.

The Complexity of the Reduction

Is it polynomial in $n = |w|$?

- The tableau has $n^k \times n^k = n^{2k}$ cells.
- Recall that $C = Q \cup \Gamma \cup \{\#\}$ and notice that C does not depend on $|w| = n$ but on the NDTM N. Each cell has $|C| = l$ variables. As l does not depend on n, the total number of variables is $O(n^{2k})$.
- Now, focus on the formulae for each of φ_{cell}, φ_{start}, φ_{move} and φ_{accept}.
- φ_{start} has a fragment for each cell in the top row. So, it is of size $O(n^k)$.
- φ_{cell}, φ_{move} and φ_{accept} each contain a fixed-size fragment of the formula for each cell of the tableau. So, they are of size $O(n^{2k})$. So, φ is polynomial in size.
- We can obviously generate φ in polynomial time as we need to do some minor manipulations with the indices i and j.