Decidability of WS1S and S1S: An Exposition

William Gasarch-U of MD

Credit Where Credit is Due

Buchi proved that WS1S was decidable.
I don't know off hand who proved S1S decidable.

Part I

We Define WS1S And Prove It's Decidable

Formulas and Sentences

(This is informal since we did not specify the language.)

Formulas and Sentences

(This is informal since we did not specify the language.)

1. A **Formula** allows variables to not be quantified over. A Formula is neither true or false. Example: $(\exists x)[x + y = 7]$.

Formulas and Sentences

(This is informal since we did not specify the language.)

1. A **Formula** allows variables to not be quantified over. A Formula is neither true or false. Example: $(\exists x)[x + y = 7]$.
2. A **Sentence** has all variables quantified over. Example: $(\forall y)(\exists x)[x + y = 7]$. So a Sentence is either true or false.

Formulas and Sentences

(This is informal since we did not specify the language.)

1. A **Formula** allows variables to not be quantified over. A Formula is neither true or false. Example: $(\exists x)[x + y = 7]$.
2. A **Sentence** has all variables quantified over. Example: $(\forall y)(\exists x)[x + y = 7]$. So a Sentence is either true or false.
   Wrong –need to also know the domain.
   $(\forall y)(\exists x)[x + y = 7]$— **T** if domain is $\mathbb{Z}$, the integers.

Formulas and Sentences

(This is informal since we did not specify the language.)

1. A **Formula** allows variables to not be quantified over. A Formula is neither true or false. Example: $(\exists x)[x + y = 7]$.
2. A **Sentence** has all variables quantified over. Example: $(\forall y)(\exists x)[x + y = 7]$. So a Sentence is either true or false.
   Wrong –need to also know the domain.
   $(\forall y)(\exists x)[x + y = 7]$— **T** if domain is $\mathbb{Z}$, the integers.
   $(\forall y)(\exists x)[x + y = 7]$— **F** if domain is $\mathbb{N}$, the naturals.

Variables and Symbols

In our lang:

Variables and Symbols

In our lang:

1. The logical symbols $\wedge$, $\neg$, ($\exists$).

Variables and Symbols

In our lang:

1. The logical symbols $\wedge$, $\neg$, $(\exists)$.
2. We use $\vee$ and $\forall$ as shorthand–can be converted to $\wedge$ and $\exists$.

Variables and Symbols

In our lang:

1. The logical symbols $\wedge$, $\neg$, ($\exists$).
2. We use $\vee$ and $\forall$ as shorthand–can be converted to $\wedge$ and $\exists$.
3. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.

Variables and Symbols

In our lang:

1. The logical symbols $\wedge$, $\neg$, $(\exists)$.
2. We use $\vee$ and $\forall$ as shorthand–can be converted to $\wedge$ and $\exists$.
3. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.
4. Variables $X, Y, Z, \ldots$ that range over finite subsets of $\mathbb{N}$.

Variables and Symbols

In our lang:

1. The logical symbols $\wedge$, $\neg$, $(\exists)$.
2. We use $\vee$ and $\forall$ as shorthand–can be converted to $\wedge$ and $\exists$.
3. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.
4. Variables $X, Y, Z, \ldots$ that range over finite subsets of $\mathbb{N}$.
5. Symbols: $<$, $\in$ (usual meaning), S (meaning $S(x) = x + 1$), $\equiv$ (congruence usual meaning), $=$ (used for both numbers and sets).

Variables and Symbols

In our lang:

1. The logical symbols $\wedge$, $\neg$, $(\exists)$.
2. We use $\vee$ and $\forall$ as shorthand–can be converted to $\wedge$ and $\exists$.
3. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.
4. Variables $X, Y, Z, \ldots$ that range over finite subsets of $\mathbb{N}$.
5. Symbols: $<$, $\in$ (usual meaning), S (meaning $S(x) = x + 1$), $\equiv$ (congruence usual meaning), $=$ (used for both numbers and sets).
6. Constants: 0,1,2,3,. . ..

Variables and Symbols

In our lang:

1. The logical symbols $\wedge$, $\neg$, $(\exists)$.
2. We use $\vee$ and $\forall$ as shorthand–can be converted to $\wedge$ and $\exists$.
3. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.
4. Variables $X, Y, Z, \ldots$ that range over finite subsets of $\mathbb{N}$.
5. Symbols: $<$, $\in$ (usual meaning), S (meaning $S(x) = x + 1$), $\equiv$ (congruence usual meaning), $=$ (used for both numbers and sets).
6. Constants: 0,1,2,3,....
7. Convention: We write $x + c$ instead of $S(S(\cdots S(x)) \cdots)$.
   NOTE $+$ is **not** in our lang.

Variables and Symbols

In our lang:

1. The logical symbols $\wedge$, $\neg$, $(\exists)$.
2. We use $\vee$ and $\forall$ as shorthand–can be converted to $\wedge$ and $\exists$.
3. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.
4. Variables $X, Y, Z, \ldots$ that range over finite subsets of $\mathbb{N}$.
5. Symbols: $<$, $\in$ (usual meaning), S (meaning $S(x) = x + 1$), $\equiv$ (congruence usual meaning), $=$ (used for both numbers and sets).
6. Constants: 0,1,2,3,. . ..
7. Convention: We write $x + c$ instead of $S(S(\cdots S(x))\cdots)$. **NOTE** $+$ is **not** in our lang.

Called WS1S: Weak Second Order Theory of One Successor.

Variables and Symbols

In our lang:

1. The logical symbols $\wedge$, $\neg$, $(\exists)$.
2. We use $\vee$ and $\forall$ as shorthand–can be converted to $\wedge$ and $\exists$.
3. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.
4. Variables $X, Y, Z, \ldots$ that range over finite subsets of $\mathbb{N}$.
5. Symbols: $<$, $\in$ (usual meaning), S (meaning $S(x) = x + 1$), $\equiv$ (congruence usual meaning), $=$ (used for both numbers and sets).
6. Constants: 0,1,2,3,....
7. Convention: We write $x + c$ instead of $S(S(\cdots S(x))\cdots)$.
   NOTE $+$ is **not** in our lang.

Called WS1S: Weak Second Order Theory of One Successor.
Weak second order means quantify over finite sets.

Examples of Formulas and Sentences

Formulas

$x \in X \wedge y + 3 \notin X$

NONE of x, y, X are quantified over, so its a formula.

Examples of Formulas and Sentences

Formulas

$x \in X \wedge y + 3 \notin X$

NONE of x, y, X are quantified over, so its a formula.

One can ask for which (x, y, X) it's TRUE.

x	y	X	T

Examples of Formulas and Sentences

Formulas

$x \in X \wedge y + 3 \notin X$

NONE of x, y, X are quantified over, so its a formula.

One can ask for which (x, y, X) it's TRUE.

x	y	X	T
0	0	$\{0\}$	T

Examples of Formulas and Sentences

Formulas

$x \in X \wedge y + 3 \notin X$

NONE of x, y, X are quantified over, so its a formula.

One can ask for which (x, y, X) it's TRUE.

x	y	X	T
0	0	$\{0\}$	T
0	1	$\{0, 1\}$	T

Examples of Formulas and Sentences

Formulas

$x \in X \wedge y + 3 \notin X$

NONE of x, y, X are quantified over, so its a formula.

One can ask for which (x, y, X) it's TRUE.

x	y	X	T
0	0	$\{0\}$	T
0	1	$\{0, 1\}$	T
0	1	$\{0, 1, 2\}$	T

Examples of Formulas and Sentences

Formulas

$x \in X \wedge y + 3 \notin X$

NONE of x, y, X are quantified over, so its a formula.

One can ask for which (x, y, X) it's TRUE.

x	y	X	T
0	0	$\{0\}$	T
0	1	$\{0, 1\}$	T
0	1	$\{0, 1, 2\}$	T
0	1	$\{0, 1, 2, 3, 4\}$	F

Examples of Formulas and Sentences

Formulas

$x \in X \wedge y + 3 \notin X$

NONE of x, y, X are quantified over, so its a formula.

One can ask for which (x, y, X) it's TRUE.

x	y	X	T
0	0	$\{0\}$	T
0	1	$\{0, 1\}$	T
0	1	$\{0, 1, 2\}$	T
0	1	$\{0, 1, 2, 3, 4\}$	F
4	7	$\{4\}$	T

Examples of Formulas and Sentences

Formulas

$x \in X \wedge y + 3 \notin X$

NONE of x, y, X are quantified over, so its a formula.

One can ask for which (x, y, X) it's TRUE.

x	y	X	T
0	0	$\{0\}$	T
0	1	$\{0, 1\}$	T
0	1	$\{0, 1, 2\}$	T
0	1	$\{0, 1, 2, 3, 4\}$	F
4	7	$\{4\}$	T
4	7	$\{7\}$	F

Examples of Formulas and Sentences

Formulas

$x \in X \wedge y + 3 \notin X$

NONE of x, y, X are quantified over, so its a formula.

One can ask for which (x, y, X) it's TRUE.

x	y	X	T
0	0	$\{0\}$	T
0	1	$\{0, 1\}$	T
0	1	$\{0, 1, 2\}$	T
0	1	$\{0, 1, 2, 3, 4\}$	F
4	7	$\{4\}$	T
4	7	$\{7\}$	F
4	7	$\{4, 7\}$	T

What Does One Successor Mean?

Our basic objects are **numbers.** View as **unary** strings, elements of 1^*. **Succ** is **Append-1.**

What Does One Successor Mean?

Our basic objects are **numbers.** View as **unary** strings, elements of 1^*. **Succ** is **Append-1.**
So 4 = (Append-1 (Append-1 (Append-1 (Append-1 0))))

What Does One Successor Mean?

Our basic objects are **numbers.** View as **unary** strings, elements of 1^*. **Succ** is **Append-1.**
So $4 =$ (Append-1 (Append-1 (Append-1 (Append-1 0))))

What IF our basic objects were **strings** in $\{0,1\}^*$? Would have **2** SUCC's: **Append-0** and **Append-1.**

What Does One Successor Mean?

Our basic objects are **numbers.** View as **unary** strings, elements of 1^*. **Succ** is **Append-1.**
So $4 =$ (Append-1 (Append-1 (Append-1 (Append-1 0))))

What IF our basic objects were **strings** in $\{0,1\}^*$? Would have **2** SUCC's: **Append-0** and **Append-1.**

WS1S Weak Second Order with **one** Successor- just one way to add to a string. Basic objects are strings of 1's.

What Does One Successor Mean?

Our basic objects are **numbers.** View as **unary** strings, elements of 1^*. **Succ** is **Append-1.**
So $4 =$ (Append-1 (Append-1 (Append-1 (Append-1 0))))

What IF our basic objects were **strings** in $\{0,1\}^*$? Would have **2** SUCC's: **Append-0** and **Append-1.**

WS1S Weak Second Order with **one** Successor- just one way to add to a string. Basic objects are strings of 1's.

WS2S Weak second order with **two** Successors- two ways to add to a string. Basic objects are strings of 0's and 1's.

What Does One Successor Mean?

Our basic objects are **numbers.** View as **unary** strings, elements of 1^*. **Succ** is **Append-1.**
So $4 =$ (Append-1 (Append-1 (Append-1 (Append-1 0))))

What IF our basic objects were **strings** in $\{0,1\}^*$? Would have **2** SUCC's: **Append-0** and **Append-1.**

WS1S Weak Second Order with **one** Successor- just one way to add to a string. Basic objects are strings of 1's.

WS2S Weak second order with **two** Successors- two ways to add to a string. Basic objects are strings of 0's and 1's.

WS2S is also decidable but we will not prove this.

Atomic Formulas

An **Atomic Formula** is:

Atomic Formulas

An **Atomic Formula** is:

1. For any $c \in \mathbb{N}$, $x = y + c$ is an Atomic Formula.

Atomic Formulas

An **Atomic Formula** is:

1. For any $c \in \mathbb{N}$, $x = y + c$ is an Atomic Formula.
2. For any $c \in \mathbb{N}$, $x < y + c$ is an Atomic Formula.

Atomic Formulas

An **Atomic Formula** is:

1. For any $c \in \mathbb{N}$, $x = y + c$ is an Atomic Formula.
2. For any $c \in \mathbb{N}$, $x < y + c$ is an Atomic Formula.
3. For any $c, d \in \mathbb{N}$, $x \equiv y + c \pmod{d}$ is an Atomic Formula.

Atomic Formulas

An **Atomic Formula** is:

1. For any $c \in \mathbb{N}$, $x = y + c$ is an Atomic Formula.
2. For any $c \in \mathbb{N}$, $x < y + c$ is an Atomic Formula.
3. For any $c, d \in \mathbb{N}$, $x \equiv y + c \pmod{d}$ is an Atomic Formula.
4. For any $c \in \mathbb{N}$, $x + c \in X$ is an Atomic Formula.

Atomic Formulas

An **Atomic Formula** is:

1. For any $c \in \mathbb{N}$, $x = y + c$ is an Atomic Formula.
2. For any $c \in \mathbb{N}$, $x < y + c$ is an Atomic Formula.
3. For any $c, d \in \mathbb{N}$, $x \equiv y + c \pmod{d}$ is an Atomic Formula.
4. For any $c \in \mathbb{N}$, $x + c \in X$ is an Atomic Formula.
5. For any $c \in \mathbb{N}$, $X = Y + c$ is an Atomic Formula.
   This means that $X = \{y + c : y \in Y\}$.

WS1S Formulas

A **WS1S Formula** is:

WS1S Formulas

A **WS1S Formula** is:

1. Any Atomic Formula is a WS1S Formula.

WS1S Formulas

A **WS1S Formula** is:

1. Any Atomic Formula is a WS1S Formula.
2. If ϕ_1, ϕ_2 are WS1S Formulas then so are

WS1S Formulas

A **WS1S Formula** is:

1. Any Atomic Formula is a WS1S Formula.
2. If ϕ_1, ϕ_2 are WS1S Formulas then so are
   2.1 $\phi_1 \wedge \phi_2$,

WS1S Formulas

A **WS1S Formula** is:

1. Any Atomic Formula is a WS1S Formula.
2. If ϕ_1, ϕ_2 are WS1S Formulas then so are
   2.1 $\phi_1 \wedge \phi_2$,
   2.2 $\phi_1 \vee \phi_2$

WS1S Formulas

A **WS1S Formula** is:

1. Any Atomic Formula is a WS1S Formula.
2. If ϕ_1, ϕ_2 are WS1S Formulas then so are
   2.1 $\phi_1 \wedge \phi_2$,
   2.2 $\phi_1 \vee \phi_2$
   2.3 $\neg\phi_1$

WS1S Formulas

A **WS1S Formula** is:

1. Any Atomic Formula is a WS1S Formula.
2. If ϕ_1, ϕ_2 are WS1S Formulas then so are
   2.1 $\phi_1 \wedge \phi_2$,
   2.2 $\phi_1 \vee \phi_2$
   2.3 $\neg\phi_1$
3. If $\phi(x_1, \ldots, x_n, X_1, \ldots, X_m)$ is a WS1S Formula then so are

WS1S Formulas

A **WS1S Formula** is:

1. Any Atomic Formula is a WS1S Formula.
2. If ϕ_1, ϕ_2 are WS1S Formulas then so are
   2.1 $\phi_1 \wedge \phi_2$,
   2.2 $\phi_1 \vee \phi_2$
   2.3 $\neg\phi_1$
3. If $\phi(x_1, \ldots, x_n, X_1, \ldots, X_m)$ is a WS1S Formula then so are
   3.1 $(\exists x_i)[\phi(x_1, \ldots, x_n, X_1, \ldots, X_m)]$

WS1S Formulas

A **WS1S Formula** is:

1. Any Atomic Formula is a WS1S Formula.
2. If ϕ_1, ϕ_2 are WS1S Formulas then so are
   2.1 $\phi_1 \wedge \phi_2$,
   2.2 $\phi_1 \vee \phi_2$
   2.3 $\neg\phi_1$
3. If $\phi(x_1, \ldots, x_n, X_1, \ldots, X_m)$ is a WS1S Formula then so are
   3.1 $(\exists x_i)[\phi(x_1, \ldots, x_n, X_1, \ldots, X_m)]$
   3.2 $(\exists X_i)[\phi(x_1, \ldots, x_n, X_1, \ldots, X_m)]$

Prenex Normal Form

A formula is in **Prenex Normal Form** if it is of the form

$$(Q_1 v_1)(Q_2 v_2)\cdots(Q_m v_m)[\phi(v_1,\ldots,v_n)]$$

where the Q_i's are quantifiers, the v_i's are either numbers or finite-set variables, and ϕ has no quantifiers. (m quantifiers, $n \geq m$ vars. This is a formula– could be vars that are not quantified over.)

Prenex Normal Form

A formula is in **Prenex Normal Form** if it is of the form

$$(Q_1 v_1)(Q_2 v_2)\cdots(Q_m v_m)[\phi(v_1,\ldots,v_n)]$$

where the Q_i's are quantifiers, the v_i's are either numbers or finite-set variables, and ϕ has no quantifiers. (m quantifiers, $n \geq m$ vars. This is a formula– could be vars that are not quantified over.)

Every formula can be put into this form using the following rules

Prenex Normal Form

A formula is in **Prenex Normal Form** if it is of the form

$$(Q_1 v_1)(Q_2 v_2) \cdots (Q_m v_m)[\phi(v_1, \ldots, v_n)]$$

where the Q_i's are quantifiers, the v_i's are either numbers or finite-set variables, and ϕ has no quantifiers. (m quantifiers, $n \geq m$ vars. This is a formula– could be vars that are not quantified over.)

Every formula can be put into this form using the following rules

1. $(\neg Qx)[\phi(x)]$ is equiv to $(Q'x)[\neg\phi(x)]$ where $Q' = \exists$ if $Q = \forall$ and $Q' = \forall$ if $Q = \exists$.

Prenex Normal Form

A formula is in **Prenex Normal Form** if it is of the form

$$(Q_1 v_1)(Q_2 v_2)\cdots(Q_m v_m)[\phi(v_1,\ldots,v_n)]$$

where the Q_i's are quantifiers, the v_i's are either numbers or finite-set variables, and ϕ has no quantifiers. (m quantifiers, $n \ge m$ vars. This is a formula– could be vars that are not quantified over.)

Every formula can be put into this form using the following rules

1. $(\neg Qx)[\phi(x)]$ is equiv to $(Q'x)[\neg\phi(x)]$ where $Q' = \exists$ if $Q = \forall$ and $Q' = \forall$ if $Q = \exists$.
2. $(Q_1 x)[\phi_1(x)] \wedge (Q_2 y)[\phi_2(y)]$ is equiv to $(Q_1 x)(Q_2 y)[\phi_1(x) \wedge \phi_2(y)]$.

Prenex Normal Form

A formula is in **Prenex Normal Form** if it is of the form

$$(Q_1 v_1)(Q_2 v_2) \cdots (Q_m v_m)[\phi(v_1, \ldots, v_n)]$$

where the Q_i's are quantifiers, the v_i's are either numbers or finite-set variables, and ϕ has no quantifiers. (m quantifiers, $n \geq m$ vars. This is a formula– could be vars that are not quantified over.)

Every formula can be put into this form using the following rules

1. $(\neg Qx)[\phi(x)]$ is equiv to $(Q'x)[\neg\phi(x)]$ where $Q' = \exists$ if $Q = \forall$ and $Q' = \forall$ if $Q = \exists$.
2. $(Q_1 x)[\phi_1(x)] \wedge (Q_2 y)[\phi_2(y)]$ is equiv to $(Q_1 x)(Q_2 y)[\phi_1(x) \wedge \phi_2(y)]$.
3. $(Q_1 x)[\phi_1(x)] \vee (Q_2 y)[\phi_2(y)]$ is equiv to $\neg((\neg Q_1 x)[\phi_1(x)] \wedge (\neg Q_2 y)[\phi_2(y)])$.

Key Definition

Def If $\phi(x_1, \ldots, x_n, X_1, \ldots, X_m)$ is a WS1S Formula then $\mathrm{TRUE}(\phi)$ is the set

$$\{(a_1, \ldots, a_n, A_1, \ldots, A_m) : \phi(a_1, \ldots, a_n, A_1, \ldots, A_m) = T\}$$

Key Definition

Def If $\phi(x_1, \ldots, x_n, X_1, \ldots, X_m)$ is a WS1S Formula then $\mathrm{TRUE}(\phi)$ is the set

$$\{(a_1, \ldots, a_n, A_1, \ldots, A_m) : \phi(a_1, \ldots, a_n, A_1, \ldots, A_m) = T\}$$

This is the set of $(a_1, \ldots, a_n, A_1, \ldots, A_m)$ that make ϕ TRUE.

Representation

We want to say that $\mathrm{TRUE}(\phi)$ is regular. Need to represent $(a_1, \ldots, a_n, A_1, \ldots, A_m)$.

Representation

We want to say that $\mathrm{TRUE}(\phi)$ is regular. Need to represent $(a_1, \ldots, a_n, A_1, \ldots, A_m)$.

We just look at (x, y, X). Use the alphabet $\{0, 1\}^3$.

Representation

We want to say that $\mathrm{TRUE}(\phi)$ is regular. Need to represent $(a_1, \ldots, a_n, A_1, \ldots, A_m)$.

We just look at (x, y, X). Use the alphabet $\{0,1\}^3$.

Below Top line and the x, y, X are not there- Visual Aid.

The triple $(3, 4, \{0, 1, 2, 4, 7\})$ is represented by

	0	1	2	3	4	5	6	7
x	0	0	0	1	$*$	$*$	$*$	$*$
y	0	0	0	0	1	$*$	$*$	$*$
X	1	1	1	0	1	0	0	1

Representation

We want to say that $\text{TRUE}(\phi)$ is regular. Need to represent $(a_1, \ldots, a_n, A_1, \ldots, A_m)$.

We just look at (x, y, X). Use the alphabet $\{0,1\}^3$.

Below Top line and the x, y, X are not there- Visual Aid.

The triple $(3, 4, \{0, 1, 2, 4, 7\})$ is represented by

	0	1	2	3	4	5	6	7
x	0	0	0	1	$*$	$*$	$*$	$*$
y	0	0	0	0	1	$*$	$*$	$*$
X	1	1	1	0	1	0	0	1

Note After we see 0001 for x we **do not care** what happens next. The *'s can be filled in with 0's or 1's and the string of symbols from $\{0,1\}^3$ above would still represent $(3, 4, \{0, 1, 2, 4, 7\})$.

Representation–More Formal

The number n is represented by $0^n1\{0,1\}^*$.

Representation–More Formal

The number n is represented by $0^n1\{0,1\}^*$.

Finite set X is represented by a string in $\{0,1\}^*$ which is its bit-vector.

Example And Our Alphabet

Consider the set

$$\{(x, y, X) : (x = y + 1) \wedge (y \in X)\}$$

We want to show that it's regular. Here is an example of how we **represent** a tuple (number,number,finite set):

	0	1	2	3	4	5	6	7
x	0	0	0	0	0	1	0	0
y	0	0	0	0	1	1	0	1
X	1	1	1	0	1	0	0	1

This string is IN our lang since $x = 5$, $y = 4$, and $X = \{0, 1, 2, 4, 7\}$.

Alphabet is $\{000, 001, 010, 011, 100, 101, 110, 111\}$ though we think of it vertically rather than horizontally.

Stupid Strings

What does

	0	1	2	3	4	5	6	7
x	0	0	0	0	0	0	0	0
y	0	0	0	0	1	1	0	1
X	1	1	1	0	1	0	0	1

represent?

Stupid Strings

What does

	0	1	2	3	4	5	6	7
x	0	0	0	0	0	0	0	0
y	0	0	0	0	1	1	0	1
X	1	1	1	0	1	0	0	1

represent?

This string is **Stupid!** There is no value for x. This string does not represent anything!

Our DFA's will have 3 kinds of states: **accept, reject,** and **stupid. Stupid** means that the string did not represent anything because it has a number-variable be all 0's. (It is fine for a set var to of all 0's- that would be the empty set.)

Key Theorem

Thm For all WS1S formulas ϕ the set $\mathrm{TRUE}(\phi)$ is regular.

We prove this by induction on the formation of a formula. If you prefer- induction on the length of a formula.

Theorem for Atomic Formulas

Lemma For all WS1S atomic formulas ϕ the set $\mathrm{TRUE}(\phi)$ is regular.
On the next few slides we give the DFA for some Atomic Formulas.
The ones we do not may be HW or on the Final.

For any $c \in \mathbb{N}$, $x = y + c$ is an Atomic Formula

For any $c \in \mathbb{N}$, $x = y + c$ is an Atomic Formula

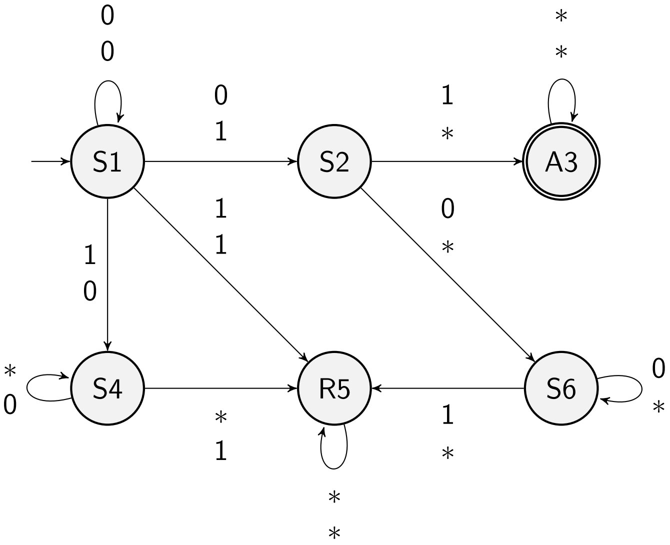
On the next slide we give the DFA for $x = y + 1$.

For any $c \in \mathbb{N}$, $x = y + c$ is an Atomic Formula

On the next slide we give the DFA for $x = y + 1$.

The DFA for $x = y + c$ is similar. Might be on a HW or the Final.

$x = y + 1$



For any $c \in \mathbb{N}$, $x < y + c$ is an Atomic Formula

For any $c \in \mathbb{N}$, $x < y + c$ is an Atomic Formula

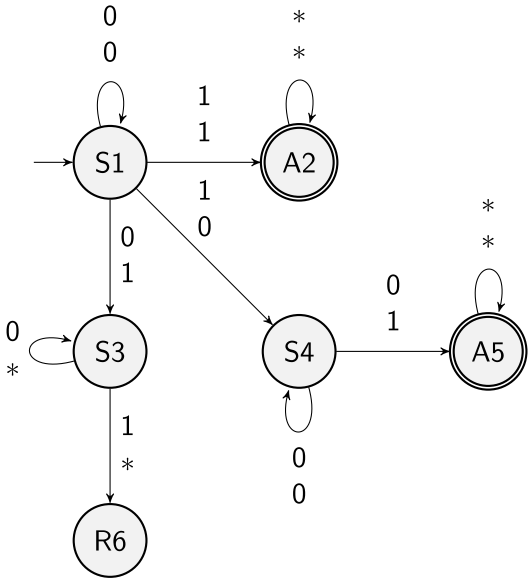
On the next slide we give the DFA for $x < y + 1$.

For any $c \in \mathbb{N}$, $x < y + c$ is an Atomic Formula

On the next slide we give the DFA for $x < y + 1$.

The DFA for $x < y + c$ is similar. Might be on a HW or the Final.

$x < y + 1$



For any $c \in \mathbb{N}$, $x + c \in X$ is an Atomic Formula

For any $c \in \mathbb{N}$, $x + c \in X$ is an Atomic Formula

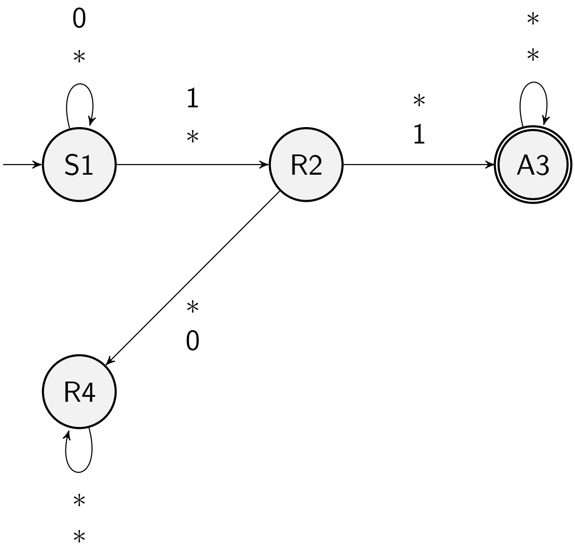
On the next slide we give the DFA for $x + 1 \in X$.

For any $c \in \mathbb{N}$, $x + c \in X$ is an Atomic Formula

On the next slide we give the DFA for $x + 1 \in X$.

The DFA for $x + c \in X$ is similar. Might be on a HW or the Final.

$x + 1 \in X$



The Remaining Atomic Formulas

We did not give DFA's for the the following Atomic Formulas:

The Remaining Atomic Formulas

We did not give DFA's for the the following Atomic Formulas:

1. For any $c, d \in \mathbb{N}$, $x \equiv y + c \pmod{d}$.

The Remaining Atomic Formulas

We did not give DFA's for the the following Atomic Formulas:

1. For any $c, d \in \mathbb{N}$, $x \equiv y + c \pmod{d}$.
2. For any $c \in \mathbb{N}$, $X = Y + c$.

The Remaining Atomic Formulas

We did not give DFA's for the the following Atomic Formulas:

1. For any $c, d \in \mathbb{N}$, $x \equiv y + c \pmod{d}$.
2. For any $c \in \mathbb{N}$, $X = Y + c$.

Getting DFA's for those atomic formulas, or special cases, might be on a HW or the Final.

Theorem for Formulas (I)

Assume true for ϕ_1, ϕ_2— so $\mathrm{TRUE}(\phi_1)$ and $\mathrm{TRUE}(\phi_2)$ are reg.

Theorem for Formulas (I)

Assume true for ϕ_1, ϕ_2— so $\mathrm{TRUE}(\phi_1)$ and $\mathrm{TRUE}(\phi_2)$ are reg.

1. $\mathrm{TRUE}(\phi_1 \wedge \phi_2) = \mathrm{TRUE}(\phi_1) \cap \mathrm{TRUE}(\phi_2)$.

Theorem for Formulas (I)

Assume true for ϕ_1, ϕ_2— so $\mathrm{TRUE}(\phi_1)$ and $\mathrm{TRUE}(\phi_2)$ are reg.

1. $\mathrm{TRUE}(\phi_1 \wedge \phi_2) = \mathrm{TRUE}(\phi_1) \cap \mathrm{TRUE}(\phi_2)$.
2. $\mathrm{TRUE}(\phi_1 \vee \phi_2) = \mathrm{TRUE}(\phi_1) \cup \mathrm{TRUE}(\phi_2)$.

Theorem for Formulas (I)

Assume true for ϕ_1, ϕ_2— so $\mathrm{TRUE}(\phi_1)$ and $\mathrm{TRUE}(\phi_2)$ are reg.

1. $\mathrm{TRUE}(\phi_1 \wedge \phi_2) = \mathrm{TRUE}(\phi_1) \cap \mathrm{TRUE}(\phi_2)$.
2. $\mathrm{TRUE}(\phi_1 \vee \phi_2) = \mathrm{TRUE}(\phi_1) \cup \mathrm{TRUE}(\phi_2)$.
3. $\mathrm{TRUE}(\neg\phi_1) = \Sigma^* - (\mathrm{TRUE}(\phi_1) \cup$ Stupid Strings).

Good News! All of the above can be shown using the Closure properties of Regular Langs.

Theorem for Formulas (I)

Assume true for ϕ_1, ϕ_2— so $\mathrm{TRUE}(\phi_1)$ and $\mathrm{TRUE}(\phi_2)$ are reg.

1. $\mathrm{TRUE}(\phi_1 \wedge \phi_2) = \mathrm{TRUE}(\phi_1) \cap \mathrm{TRUE}(\phi_2)$.
2. $\mathrm{TRUE}(\phi_1 \vee \phi_2) = \mathrm{TRUE}(\phi_1) \cup \mathrm{TRUE}(\phi_2)$.
3. $\mathrm{TRUE}(\neg\phi_1) = \Sigma^* - (\mathrm{TRUE}(\phi_1) \cup$ Stupid Strings$)$.

Good News! All of the above can be shown using the Closure properties of Regular Langs.

Caveat Must be done carefully because of the stupid states. (Stupid is as stupid does. Name that movie reference!)

Theorem for Formulas (I)

Assume true for ϕ_1, ϕ_2— so $\mathrm{TRUE}(\phi_1)$ and $\mathrm{TRUE}(\phi_2)$ are reg.

1. $\mathrm{TRUE}(\phi_1 \wedge \phi_2) = \mathrm{TRUE}(\phi_1) \cap \mathrm{TRUE}(\phi_2)$.
2. $\mathrm{TRUE}(\phi_1 \vee \phi_2) = \mathrm{TRUE}(\phi_1) \cup \mathrm{TRUE}(\phi_2)$.
3. $\mathrm{TRUE}(\neg\phi_1) = \Sigma^* - (\mathrm{TRUE}(\phi_1) \cup$ Stupid Strings).

Good News! All of the above can be shown using the Closure properties of Regular Langs.

Caveat Must be done carefully because of the stupid states. (Stupid is as stupid does. Name that movie reference!)

Next slides for what to do about quantifiers.

Theorem for Formulas (II)

$\mathrm{TRUE}(\phi(x_1, \ldots, x_n, X_1, \ldots, X_m))$ is regular.
We want $\mathrm{TRUE}((\exists x_1)[\phi(x_1, \ldots, x_n, X_1, \ldots, X_m)])$ is regular.
Ideas?

Theorem for Formulas (II)

$\mathrm{TRUE}(\phi(x_1, \ldots, x_n, X_1, \ldots, X_m))$ is regular.
We want $\mathrm{TRUE}((\exists x_1)[\phi(x_1, \ldots, x_n, X_1, \ldots, X_m)])$ is regular.
Ideas?
Use nondeterminism.
Will show you in class.

DFA Decidability Theorem

We need the following easy theorem:
Thm The following problem is decidable: given a DFA determine if **there exists** a string it accepts.

DFA Decidability Theorem Proof

Thm The following problem is decidable: given a DFA determine if **there exists** a string it accepts.
Might be on HW.

Decidability of WS1S

Thm WS1S is Decidable.
Proof

Decidability of WS1S

Thm WS1S is Decidable.
Proof

1. Given a **sentence** in WS1S put it into the form

$$(Q_1X_1)\cdots(Q_nX_n)(Q_{n+1}x_1)\cdots(Q_{n+m}x_m)[\phi(x_1,\ldots,x_m,X_1,\ldots,X_n)]$$

Decidability of WS1S

Thm WS1S is Decidable.
Proof

1. Given a **sentence** in WS1S put it into the form

   $$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

2. Assume $Q_1 = \exists$. (If not then negate and negate answer.)

Decidability of WS1S

Thm WS1S is Decidable.
Proof

1. Given a **sentence** in WS1S put it into the form

$$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

2. Assume $Q_1 = \exists$. (If not then negate and negate answer.)
3. View as $(\exists X)[\phi(X)]$, a **formula** with **one** free var.

Decidability of WS1S

Thm WS1S is Decidable.
Proof

1. Given a **sentence** in WS1S put it into the form

$$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

2. Assume $Q_1 = \exists$. (If not then negate and negate answer.)
3. View as $(\exists X)[\phi(X)]$, a **formula** with **one** free var.
4. Construct DFA M for $\{X : \phi(X) \text{ is true}\}$.

Decidability of WS1S

Thm WS1S is Decidable.
Proof

1. Given a **sentence** in WS1S put it into the form

   $$(Q_1X_1)\cdots(Q_nX_n)(Q_{n+1}x_1)\cdots(Q_{n+m}x_m)[\phi(x_1,\ldots,x_m,X_1,\ldots,X_n)]$$

2. Assume $Q_1 = \exists$. (If not then negate and negate answer.)
3. View as $(\exists X)[\phi(X)]$, a **formula** with **one** free var.
4. Construct DFA M for $\{X : \phi(X) \text{ is true}\}$.
5. Test if $L(M) = \emptyset$.

Decidability of WS1S

Thm WS1S is Decidable.
Proof

1. Given a **sentence** in WS1S put it into the form

   $$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

2. Assume $Q_1 = \exists$. (If not then negate and negate answer.)
3. View as $(\exists X)[\phi(X)]$, a **formula** with **one** free var.
4. Construct DFA M for $\{X : \phi(X) \text{ is true}\}$.
5. Test if $L(M) = \emptyset$.
6. If $L(M) \neq \emptyset$ then $(\exists X)[\phi(X)]$ is TRUE.

Decidability of WS1S

Thm WS1S is Decidable.
Proof

1. Given a **sentence** in WS1S put it into the form

   $$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

2. Assume $Q_1 = \exists$. (If not then negate and negate answer.)
3. View as $(\exists X)[\phi(X)]$, a **formula** with **one** free var.
4. Construct DFA M for $\{X : \phi(X) \text{ is true}\}$.
5. Test if $L(M) = \emptyset$.
6. If $L(M) \neq \emptyset$ then $(\exists X)[\phi(X)]$ is TRUE.
   If $L(M) = \emptyset$ then $(\exists X)[\phi(X)]$ is FALSE.

An Example

We will do the following **together.**

An Example

We will do the following **together.**

$$(\exists X)(\exists x)(\forall y)[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)].$$

An Example

We will do the following **together.**

$$(\exists X)(\exists x)(\forall y)[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)].$$

We need DFA's for the following:

An Example

We will do the following **together.**

$$(\exists X)(\exists x)(\forall y)[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)].$$

We need DFA's for the following:

1. $\{(x, y, X) : x \in X\}$

An Example

We will do the following **together.**

$$(\exists X)(\exists x)(\forall y)[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)].$$

We need DFA's for the following:

1. $\{(x, y, X) : x \in X\}$
2. $\{(x, y, X) : x \geq 2\}$

An Example

We will do the following **together.**

$$(\exists X)(\exists x)(\forall y)[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)].$$

We need DFA's for the following:

1. $\{(x, y, X) : x \in X\}$
2. $\{(x, y, X) : x \geq 2\}$
3. $\{(x, y, X) : y > x\}$

An Example

We will do the following **together.**

$$(\exists X)(\exists x)(\forall y)[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)].$$

We need DFA's for the following:

1. $\{(x, y, X) : x \in X\}$
2. $\{(x, y, X) : x \geq 2\}$
3. $\{(x, y, X) : y > x\}$
4. $\{(x, y, X) : y \notin X\}$

Atomic Formulas we Need

We get DFA's for the following in order, using the prior ones to get the later ones.

Atomic Formulas we Need

We get DFA's for the following in order, using the prior ones to get the later ones.

1. $\{(x, y, X) : x \in X \wedge x \geq 2\}$

Atomic Formulas we Need

We get DFA's for the following in order, using the prior ones to get the later ones.

1. $\{(x, y, X) : x \in X \wedge x \geq 2\}$
2. $\{(x, y, X) : y > x \vee y \notin X\})$

Atomic Formulas we Need

We get DFA's for the following in order, using the prior ones to get the later ones.

1. $\{(x, y, X) : x \in X \wedge x \geq 2\}$
2. $\{(x, y, X) : y > x \vee y \notin X\})$
3. $\{(x, y, X) : x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)\})$

Atomic Formulas we Need

We get DFA's for the following in order, using the prior ones to get the later ones.

1. $\{(x, y, X) : x \in X \wedge x \geq 2\}$
2. $\{(x, y, X) : y > x \vee y \notin X\})$
3. $\{(x, y, X) : x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)\})$
4. $\{(x, y, X) : \neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}$

Atomic Formulas we Need

We get DFA's for the following in order, using the prior ones to get the later ones.

1. $\{(x, y, X) : x \in X \wedge x \geq 2\}$
2. $\{(x, y, X) : y > x \vee y \notin X\})$
3. $\{(x, y, X) : x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)\})$
4. $\{(x, y, X) : \neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}$

Note No De Morgans Law–we complement the DFA.

Atomic Formulas we Need

We have a DFA for

$$\{(x, y, X) : \neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}$$

Atomic Formulas we Need

We have a DFA for

$$\{(x, y, X) : \neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}$$

We get DFA's for the following by projection and complementation.

Atomic Formulas we Need

We have a DFA for

$$\{(x, y, X) : \neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}$$

We get DFA's for the following by projection and complementation.

1. $\{(x, X) : (\exists y)\neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}$

Atomic Formulas we Need

We have a DFA for

$$\{(x, y, X) : \neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}$$

We get DFA's for the following by projection and complementation.

1. $\{(x, X) : (\exists y)\neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}$
2. $\{(x, X) : \neg(\exists y)\neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}$

Atomic Formulas we Need

We have a DFA for

$$\{(x, y, X) : \neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}$$

We get DFA's for the following by projection and complementation.

1. $\{(x, X) : (\exists y)\neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}$
2. $\{(x, X) : \neg(\exists y)\neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}$
3. $\{X : (\exists x)\neg(\exists y)\neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}$

The Finale!

Take the DFA for

$$\{X : (\exists x)\neg(\exists y)\neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}.$$

The Finale!

Take the DFA for

$$\{X : (\exists x)\neg(\exists y)\neg[x \in X \land x \geq 2 \land (y > x \lor y \notin X)]\}.$$

Test it –does there exist a string it accepts?

The Finale!

Take the DFA for

$$\{X : (\exists x)\neg(\exists y)\neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}.$$

Test it –does there exist a string it accepts?

If YES then the original sentence is TRUE.

The Finale!

Take the DFA for

$$\{X : (\exists x)\neg(\exists y)\neg[x \in X \wedge x \geq 2 \wedge (y > x \vee y \notin X)]\}.$$

Test it –does there exist a string it accepts?

If YES then the original sentence is TRUE.
If NO then the original sentence is FALSE.

Complexity of the Decision Procedure

Given a sentence

$$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

How long will the procedure above take in the worst case?:

Complexity of the Decision Procedure

Given a sentence

$$(Q_1 X_1)\cdots(Q_n X_n)(Q_{n+1} x_1)\cdots(Q_{n+m} x_m)[\phi(x_1,\ldots,x_m,X_1,\ldots,X_n)]$$

How long will the procedure above take in the worst case?:
$2^{2^{\cdots n}}$ steps since we do n nondet to det transformations.
Vote

Complexity of the Decision Procedure

Given a sentence

$$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

How long will the procedure above take in the worst case?:
$2^{2^{\cdots n}}$ steps since we do n nondet to det transformations.

Vote

1. There are much better algorithms.

Complexity of the Decision Procedure

Given a sentence

$$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

How long will the procedure above take in the worst case?:
$2^{2^{\cdots n}}$ steps since we do n nondet to det transformations.

Vote

1. There are much better algorithms.
2. $2^{2^{\cdots n}}$ steps is provably the best you can do (roughly).

Complexity of the Decision Procedure

Given a sentence

$$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

How long will the procedure above take in the worst case?:
$2^{2^{\cdots n}}$ steps since we do n nondet to det transformations.

Vote

1. There are much better algorithms.
2. $2^{2^{\cdots n}}$ steps is provably the best you can do (roughly).
3. Complexity of dec of WS1S is unknown to science!

Complexity of the Decision Procedure

Given a sentence

$$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

How long will the procedure above take in the worst case?:
$2^{2^{\cdots n}}$ steps since we do n nondet to det transformations.

Vote

1. There are much better algorithms.
2. $2^{2^{\cdots n}}$ steps is provably the best you can do (roughly).
3. Complexity of dec of WS1S is unknown to science!

And the answer is: Can do better: $\mathbf{2^{2^{n^3 \log n}}}$. **This is provably the best you can do (roughly).**

Anything Interesting STATABLE IN WS1S?

Are there interesting problems that can be STATED in WS1S?
VOTE:

Anything Interesting STATABLE IN WS1S?

Are there interesting problems that can be STATED in WS1S?
VOTE:

1. **YES**

Anything Interesting STATABLE IN WS1S?

Are there interesting problems that can be STATED in WS1S?
VOTE:

1. **YES**
2. **NO**

Anything Interesting STATABLE IN WS1S?

Are there interesting problems that can be STATED in WS1S?
VOTE:

1. **YES**
2. **NO**

Depends what you find interesting.

Anything Interesting STATABLE IN WS1S?

Are there interesting problems that can be STATED in WS1S?
VOTE:

1. **YES**
2. **NO**

Depends what you find interesting.

YES Extensions of WS1S are used in low-level verification of code fragments. The MONA group has coded this up and used it, though their code uses MANY tricks to speed up the program in MOST cases.

Anything Interesting STATABLE IN WS1S?

Are there interesting problems that can be STATED in WS1S?
VOTE:

1. **YES**
2. **NO**

Depends what you find interesting.

YES Extensions of WS1S are used in low-level verification of code fragments. The MONA group has coded this up and used it, though their code uses MANY tricks to speed up the program in MOST cases.

NO There are no interesting MATH problems that can be expressed in WS1S.

PRESBURGER ARITHMETIC

In our lang

PRESBURGER ARITHMETIC

In our lang

1. The logical symbols $\wedge$, $\vee$, $\neg$, $(\exists)$, $(\forall)$.

PRESBURGER ARITHMETIC

In our lang

1. The logical symbols $\wedge$, $\vee$, $\neg$, $(\exists)$, $(\forall)$.
2. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.

PRESBURGER ARITHMETIC

In our lang

1. The logical symbols $\wedge$, $\vee$, $\neg$, $(\exists)$, $(\forall)$.
2. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.
3. Symbols: $<$, $+$. Constants: 0,1,2,3,. . ..

PRESBURGER ARITHMETIC

In our lang

1. The logical symbols $\wedge$, $\vee$, $\neg$, $(\exists)$, $(\forall)$.
2. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.
3. Symbols: $<$, $+$. Constants: 0,1,2,3,. . ..

Terms and Formulas:

PRESBURGER ARITHMETIC

In our lang

1. The logical symbols $\wedge$, $\vee$, $\neg$, $(\exists)$, $(\forall)$.
2. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.
3. Symbols: $<$, $+$. Constants: 0,1,2,3,. . ..

Terms and Formulas:

1. Any variable or constant is a term.

PRESBURGER ARITHMETIC

In our lang

1. The logical symbols $\wedge$, $\vee$, $\neg$, $(\exists)$, $(\forall)$.
2. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.
3. Symbols: $<$, $+$. Constants: 0,1,2,3,. . ..

Terms and Formulas:

1. Any variable or constant is a term.
2. t_1, t_2 terms then $t_1 + t_2$ is term.

PRESBURGER ARITHMETIC

In our lang

1. The logical symbols $\wedge$, $\vee$, $\neg$, $(\exists)$, $(\forall)$.
2. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.
3. Symbols: $<$, $+$. Constants: 0,1,2,3,$\ldots$.

Terms and Formulas:

1. Any variable or constant is a term.
2. t_1, t_2 terms then $t_1 + t_2$ is term.
3. t_1, t_2 terms then $t_1 = t_2$, $t_1 < t_2$ are atomic formulas.

PRESBURGER ARITHMETIC

In our lang

1. The logical symbols $\wedge$, $\vee$, $\neg$, $(\exists)$, $(\forall)$.
2. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.
3. Symbols: $<$, $+$. Constants: 0,1,2,3,….

Terms and Formulas:

1. Any variable or constant is a term.
2. t_1, t_2 terms then $t_1 + t_2$ is term.
3. t_1, t_2 terms then $t_1 = t_2$, $t_1 < t_2$ are atomic formulas.
4. Other formulas in usual way: $\wedge$, $\vee$, $\neg$, $(\exists)$, $(\forall)$.

PRESBURGER ARITHMETIC

In our lang

1. The logical symbols $\wedge$, $\vee$, $\neg$, $(\exists)$, $(\forall)$.
2. Variables $x, y, z, \ldots$ that range over $\mathbb{N}$.
3. Symbols: $<$, $+$. Constants: 0,1,2,3,....

Terms and Formulas:

1. Any variable or constant is a term.
2. t_1, t_2 terms then $t_1 + t_2$ is term.
3. t_1, t_2 terms then $t_1 = t_2$, $t_1 < t_2$ are atomic formulas.
4. Other formulas in usual way: $\wedge$, $\vee$, $\neg$, $(\exists)$, $(\forall)$.

Presb Arith is decidable by TRANSFORMING Pres Arith Sentences into WS1S sentences.
Presb Arithmetic has been used in Code Optimization (using a better dec procedure than reducing to WS1S).

S1S

PART II OF THIS TALK:
WE DEFINE S1S AND PROVE IT'S DECIDABLE

What is S1S?

What's The Same? We use the same symbols and define formulas and sentences the same way

What is S1S?

What's The Same? We use the same symbols and define formulas and sentences the same way

What's Different? We interpret the set variables as ranging over ANY set of naturals, including infinite ones.

What is S1S?

What's The Same? We use the same symbols and define formulas and sentences the same way

What's Different? We interpret the set variables as ranging over ANY set of naturals, including infinite ones.

The following sentence is TRUE in S1S but FALSE in WS1S

$$(\exists X)(\forall x)(\exists y)[y > x \wedge y \in X]$$

It says that there exists an infinite set.

What is S1S?

What's The Same? We use the same symbols and define formulas and sentences the same way

What's Different? We interpret the set variables as ranging over ANY set of naturals, including infinite ones.

The following sentence is TRUE in S1S but FALSE in WS1S

$$(\exists X)(\forall x)(\exists y)[y > x \land y \in X]$$

It says that there exists an infinite set.

Question Can we still use finite automata?

Essence of WS1S proof

Essence of WS1S Proof

Essence of WS1S proof

Essence of WS1S Proof

1. Reg langs closed: UNION, INTER, COMP, PROJ.

Essence of WS1S proof

Essence of WS1S Proof

1. Reg langs closed: UNION, INTER, COMP, PROJ.
2. Emptyness problem for DFA's is decidable.

Essence of WS1S proof

Essence of WS1S Proof

1. Reg langs closed: UNION, INTER, COMP, PROJ.
2. Emptyness problem for DFA's is decidable.

KEY We never actually RAN a DFA on any string.

Essence of WS1S proof

Essence of WS1S Proof

1. Reg langs closed: UNION, INTER, COMP, PROJ.
2. Emptyness problem for DFA's is decidable.

KEY We never actually RAN a DFA on any string.
Def A ***B*-NFA** is an NFA. If $x \in \Sigma^\omega$ then x is accepted by B-NFA M if there is a path such that $M(x)$ hits a final state inf often.
Good News: (PROVE IN GROUPS)

Essence of WS1S proof

Essence of WS1S Proof

1. Reg langs closed: UNION, INTER, COMP, PROJ.
2. Emptyness problem for DFA's is decidable.

KEY We never actually RAN a DFA on any string.
Def A ***B*-NFA** is an NFA. If $x \in \Sigma^\omega$ then x is accepted by B-NFA M if there is a path such that $M(x)$ hits a final state inf often.
Good News: (PROVE IN GROUPS)

1. B-reg closed: UNION, INTER, PROJ.

Essence of WS1S proof

Essence of WS1S Proof

1. Reg langs closed: UNION, INTER, COMP, PROJ.
2. Emptyness problem for DFA's is decidable.

KEY We never actually RAN a DFA on any string.

Def A ***B*-NFA** is an NFA. If $x \in \Sigma^{\omega}$ then x is accepted by B-NFA M if there is a path such that $M(x)$ hits a final state inf often.

Good News: (PROVE IN GROUPS)

1. B-reg closed: UNION, INTER, PROJ.
2. Emptyness problem for B-NFA's is decidable.

Essence of WS1S proof

Essence of WS1S Proof

1. Reg langs closed: UNION, INTER, COMP, PROJ.
2. Emptyness problem for DFA's is decidable.

KEY We never actually RAN a DFA on any string.
Def A ***B*-NFA** is an NFA. If $x \in \Sigma^\omega$ then x is accepted by B-NFA M if there is a path such that $M(x)$ hits a final state inf often.
Good News: (PROVE IN GROUPS)

1. B-reg closed: UNION, INTER, PROJ.
2. Emptyness problem for B-NFA's is decidable.

Need B-reg closed under complementation.

GOOD NEWS EVERYONE!

Good News B-reg **is** closed under Complementation.

GOOD NEWS EVERYONE!

Good News B-reg **is** closed under Complementation.
Good News That is **all** we need to get S1S decidable.

GOOD NEWS EVERYONE!

Good News B-reg **is** closed under Complementation.
Good News That is **all** we need to get S1S decidable.
Good News It's the **only** hard step!

GOOD NEWS EVERYONE!

Good News B-reg **is** closed under Complementation.
Good News That is **all** we need to get S1S decidable.
Good News It's the **only** hard step!
Good News We are **not** going to prove it.

GOOD NEWS EVERYONE!

Good News B-reg **is** closed under Complementation.
Good News That is **all** we need to get S1S decidable.
Good News It's the **only** hard step!
Good News We are **not** going to prove it.
Odd News Proof Uses

GOOD NEWS EVERYONE!

Good News B-reg **is** closed under Complementation.
Good News That is **all** we need to get S1S decidable.
Good News It's the **only** hard step!
Good News We are **not** going to prove it.
Odd News Proof Uses **Ramsey Theory,** yet I never proved it in my Ramsey Theory course.

B-Reg and Mu-Reg

Def A ***Mu*-aut** M is a $(Q, \Sigma, \delta, s, \mathcal{F})$ where Q, Σ, δ, s are as usual but $\mathcal{F} \subseteq 2^Q$.

B-Reg and Mu-Reg

Def A ***Mu*-aut** M is a $(Q, \Sigma, \delta, s, \mathcal{F})$ where Q, Σ, δ, s are as usual but $\mathcal{F} \subseteq 2^Q$.

That is $\mathcal{F}$ is a **set** of sets of states.

B-Reg and Mu-Reg

Def A ***Mu*-aut** M is a $(Q, \Sigma, \delta, s, \mathcal{F})$ where Q, Σ, δ, s are as usual but $\mathcal{F} \subseteq 2^Q$.

That is $\mathcal{F}$ is a **set** of sets of states.

M accepts $x \in \Sigma^\omega$ if when you run $M(x)$ the **set of states visited inf often** is in $\mathcal{F}$.

B-Reg and Mu-Reg

Def A ***Mu*-aut** M is a $(Q, \Sigma, \delta, s, \mathcal{F})$ where Q, Σ, δ, s are as usual but $\mathcal{F} \subseteq 2^Q$.

That is $\mathcal{F}$ is a **set** of sets of states.

M accepts $x \in \Sigma^\omega$ if when you run $M(x)$ the **set of states visited inf often** is in $\mathcal{F}$.

Easy (IN GROUPS) Mu-reg Closed: UNION, INTER, COMP.

PLAN

Recap and Plan

PLAN

Recap and Plan

- B-reg easily closed: $\cup$, $\cap$, PROJ, but COMPLEMENT hard.

PLAN

Recap and Plan

- *B*-reg easily closed: $\cup$, $\cap$, PROJ, but COMPLEMENT hard.
- *Mu*-reg easily closed: $\cup$, $\cap$, COMPLEMENT. But PROJ hard.

PLAN

Recap and Plan

- B-reg easily closed: $\cup$, $\cap$, PROJ, but COMPLEMENT hard.
- Mu-reg easily closed: $\cup$, $\cap$, COMPLEMENT. But PROJ hard.
- How to prove? Show B-reg $=$ Mu-reg.

DECIDABILITY OF S1S

Thm S1S is Decidable.

DECIDABILITY OF S1S

Thm S1S is Decidable.
Pf

DECIDABILITY OF S1S

Thm S1S is Decidable.
Pf

1. Given a SENTENCE in S1S put it into the form

$$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

DECIDABILITY OF S1S

Thm S1S is Decidable.
Pf

1. Given a SENTENCE in S1S put it into the form

$$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

2. Assume $Q_1 = \exists$. (If not then negate and negate answer.)

DECIDABILITY OF S1S

Thm S1S is Decidable.
Pf

1. Given a SENTENCE in S1S put it into the form

$$(Q_1X_1)\cdots(Q_nX_n)(Q_{n+1}x_1)\cdots(Q_{n+m}x_m)[\phi(x_1,\ldots,x_m,X_1,\ldots,X_n)]$$

2. Assume $Q_1 = \exists$. (If not then negate and negate answer.)
3. View as $(\exists X)[\phi(X)]$, a FORMULA with ONE free var.

DECIDABILITY OF S1S

Thm S1S is Decidable.

Pf

1. Given a SENTENCE in S1S put it into the form

   $$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

2. Assume $Q_1 = \exists$. (If not then negate and negate answer.)
3. View as $(\exists X)[\phi(X)]$, a FORMULA with ONE free var.
4. Construct B-NFA M for $\{X : \phi(X) \text{ is true}\}$.

DECIDABILITY OF S1S

Thm S1S is Decidable.
Pf

1. Given a SENTENCE in S1S put it into the form

$$(Q_1X_1)\cdots(Q_nX_n)(Q_{n+1}x_1)\cdots(Q_{n+m}x_m)[\phi(x_1,\ldots,x_m,X_1,\ldots,X_n)]$$

2. Assume $Q_1 = \exists$. (If not then negate and negate answer.)
3. View as $(\exists X)[\phi(X)]$, a FORMULA with ONE free var.
4. Construct B-NFA M for $\{X : \phi(X) \text{ is true}\}$.
5. Test if $L(M) = \emptyset$.

DECIDABILITY OF S1S

Thm S1S is Decidable.
Pf

1. Given a SENTENCE in S1S put it into the form

$$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

2. Assume $Q_1 = \exists$. (If not then negate and negate answer.)
3. View as $(\exists X)[\phi(X)]$, a FORMULA with ONE free var.
4. Construct B-NFA M for $\{X : \phi(X) \text{ is true}\}$.
5. Test if $L(M) = \emptyset$.
6. If $L(M) \neq \emptyset$ then $(\exists X)[\phi(X)]$ is TRUE.

DECIDABILITY OF S1S

Thm S1S is Decidable.
Pf

1. Given a SENTENCE in S1S put it into the form

   $$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

2. Assume $Q_1 = \exists$. (If not then negate and negate answer.)
3. View as $(\exists X)[\phi(X)]$, a FORMULA with ONE free var.
4. Construct B-NFA M for $\{X : \phi(X) \text{ is true}\}$.
5. Test if $L(M) = \emptyset$.
6. If $L(M) \neq \emptyset$ then $(\exists X)[\phi(X)]$ is TRUE.
   If $L(M) = \emptyset$ then $(\exists X)[\phi(X)]$ is FALSE.

COMPLEXITY OF THE DECISION PROCEDURE

Given a sentence

$$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

COMPLEXITY OF THE DECISION PROCEDURE

Given a sentence

$$(Q_1 X_1) \cdots (Q_n X_n)(Q_{n+1} x_1) \cdots (Q_{n+m} x_m)[\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)]$$

How long will the procedure above take in the worst case?
$2^{2^{\cdots n}}$ steps since we do n nondet to det transformations.

Anything Interesting STATABLE IN S1S?

Are there interesting problems that can be STATED in S1S?
YES Verification of programs that are supposed to run forever like operating systems. Verification of security protocols.

Anything Interesting STATABLE IN S1S?

Are there interesting problems that can be STATED in S1S?

YES Verification of programs that are supposed to run forever like operating systems. Verification of security protocols.

NO There are no interesting MATH problems that can be expressed in S1S.

Extension

WS1S, S1S are about strings 0^*1 and sets of such strings.

Extension

WS1S, S1S are about strings 0^*1 and sets of such strings.

WS2S, S2S are about strings $\{0,1\}^*$ and sets of such strings.

Extension

WS1S, S1S are about strings 0^*1 and sets of such strings.

WS2S, S2S are about strings $\{0,1\}^*$ and sets of such strings.

CAN Anything Interesting Be Stated in WS2S or S2S?

Extension

WS1S, S1S are about strings 0^*1 and sets of such strings.

WS2S, S2S are about strings $\{0,1\}^*$ and sets of such strings.

CAN Anything Interesting Be Stated in WS2S or S2S?

WS2S: YES for verification, no for mathematics.

Extension

WS1S, S1S are about strings 0^*1 and sets of such strings.

WS2S, S2S are about strings $\{0,1\}^*$ and sets of such strings.

CAN Anything Interesting Be Stated in WS2S or S2S?

WS2S: YES for verification, no for mathematics.

S2S: YES for mathematics (finally!). Verification- probably.

Extension

WS1S, S1S are about strings 0^*1 and sets of such strings.

WS2S, S2S are about strings $\{0,1\}^*$ and sets of such strings.

CAN Anything Interesting Be Stated in WS2S or S2S?

WS2S: YES for verification, no for mathematics.

S2S: YES for mathematics (finally!). Verification- probably.

I do not think S2S has ever been coded up.

Extension

WS1S, S1S are about strings 0^*1 and sets of such strings.

WS2S, S2S are about strings $\{0,1\}^*$ and sets of such strings.

CAN Anything Interesting Be Stated in WS2S or S2S?

WS2S: YES for verification, no for mathematics.

S2S: YES for mathematics (finally!). Verification- probably.

I do not think S2S has ever been coded up.
Coding it up might be a good project.

Extension

WS1S, S1S are about strings 0^*1 and sets of such strings.

WS2S, S2S are about strings $\{0,1\}^*$ and sets of such strings.

CAN Anything Interesting Be Stated in WS2S or S2S?

WS2S: YES for verification, no for mathematics.

S2S: YES for mathematics (finally!). Verification- probably.

I do not think S2S has ever been coded up.
Coding it up might be a good project. Or not.

ω-Reg

Def A language L is **ω-reg** if there exists regular langs $U_1, U_2, \ldots, U_n,\ V_1, V_2, \ldots, V_n$ such that

$$L = \bigcup_{i=1}^{n} U_i V_i^{\omega}.$$

Thm B-reg $= \omega$-reg

Work with Neighbors

Lim-Reg

Def

Lim-Reg

Def

1. Let $V \subseteq \Sigma^*$.

$$\text{ioPrefix(V)} = \{x = \sigma_1\sigma_2\cdots \in \Sigma^\omega : (\exists^\infty i)[\sigma_1\cdots\sigma_i \in V]\}$$

Lim-Reg

Def

1. Let $V \subseteq \Sigma^*$.

$$\mathrm{ioPrefix(V)} = \{x = \sigma_1\sigma_2 \cdots \in \Sigma^\omega : (\exists^\infty i)[\sigma_1 \cdots \sigma_i \in V]\}$$

2. A language L is **ioPrefix-reg** if there exists regular langs $U_1, U_2, \ldots, U_n, V_1, V_2, \ldots, V_n$ such that

$$L = \bigcup_{i=1}^{n} U_i \cdot \mathrm{ioPrefix}(V)$$

FILL OUT COURSE EVALS for ALL YOUR COURSES!!!

William Gasarch-U of MD