Quantitative Logics

George Rahonis

Department of Mathematics
Aristotle University of Thessaloniki, Greece

RISC - Formal Methods seminar
Linz, May 26, 2010

Motivation

Why do we need a quantitative setup?

- Analysis of Quantitative Systems
  - Probabilistic systems
  - Minimization of costs
  - Maximization of rewards
  - Computation of reliability
  - Optimization of energy consumption
- Natural language processing
- Speech recognition
- Digital image compression
- Fuzzy systems

Motivation

Models

- Probabilistic automata
- Transition systems with costs
- Transition systems with rewards
- Transducers with weights
- Fuzzy automata
- ...

Motivation

Models

- Probabilistic automata
- Transition systems with costs
- Transition systems with rewards
- Transducers with weights
- Fuzzy automata
- . . .

} Weighted Automata

Motivation

Weighted automata introduced by M. Schützenberger (1961)

Handbook of Weighted Automata,
Manfred Droste, Werner Kuich, and Heiko Vogler eds.,
Monographs in Theoretical Computer Science, An EATCS Series, Springer 2009.

Quantitative analysis: the specification languages (MSO, LTL, CTL, ...) should be also quantitative

Quantitative Monadic Second Order (MSO) logic

State of the art

Weighted MSO logic over:

- finite words   Droste & Gastin 2005, 2009,
- infinite words   Droste & R 2006,
- finite and infinite words with discounting   Droste & Rahonis 2007,
- finite trees   Droste & Vogler 2006,
- infinite trees   R 2007,
- finite and infinite trees with discounting   Mandrali & R 2009,
- unranked trees   Droste & Vogler 2009,
- pictures   Fichtner 2006,
- texts   Mathissen 2007,
- traces   Meinecke 2006,
- distributed systems   Bollig & Meinecke 2007,

Multi-valued MSO logic over words and trees   Droste, Kuich & R 2008,

...

State of the art

Multi-valued LTL Kupferman & Lustig 2007,

Weighted LTL:

- with discounting Mandrali 2010,
- extended with discounting R 2009,
- over arbitrary semirings Mandrali & R (in progress),

. . .

Overview

- Recall finite automata over finite and infinite words

Overview

- Recall finite automata over finite and infinite words
- MSO logic

Overview

- Recall finite automata over finite and infinite words
- MSO logic
- Weighted automata over finite words

Overview

- Recall finite automata over finite and infinite words
- MSO logic
- Weighted automata over finite words
- Weighted Büchi automata

Overview

- Recall finite automata over finite and infinite words
- MSO logic
- Weighted automata over finite words
- Weighted Büchi automata
- Weighted MSO logic

Overview

- Recall finite automata over finite and infinite words
- MSO logic
- Weighted automata over finite words
- Weighted Büchi automata
- Weighted MSO logic
    - . . . over finite words

Overview

- Recall finite automata over finite and infinite words
- MSO logic
- Weighted automata over finite words
- Weighted Büchi automata
- Weighted MSO logic
    - . . . over finite words
    - . . . over infinite words

Overview

- Recall finite automata over finite and infinite words
- MSO logic
- Weighted automata over finite words
- Weighted Büchi automata
- Weighted MSO logic
  - ...over finite words
  - ...over infinite words
- Weighted automata and MSO logic with discounting

Overview

- Recall finite automata over finite and infinite words
- MSO logic
- Weighted automata over finite words
- Weighted Büchi automata
- Weighted MSO logic
    - ...over finite words
    - ...over infinite words
- Weighted automata and MSO logic with discounting
- Weighted LTL with discounting

Overview

- Recall finite automata over finite and infinite words
- MSO logic
- Weighted automata over finite words
- Weighted Büchi automata
- Weighted MSO logic
  - ...over finite words
  - ...over infinite words
- Weighted automata and MSO logic with discounting
- Weighted LTL with discounting
- Open problems and future work

Words

- an *alphabet* A is a finite set

Words

- an *alphabet* A is a finite set
- $A^* = \{\varepsilon\} \cup \{a_0 \ldots a_{n-1} \mid n > 1, a_0, \ldots, a_{n-1} \in A\}$: the set of all *finite words* over A (free monoid generated by A)

Words

- an *alphabet* A is a finite set
- $A^* = \{\varepsilon\} \cup \{a_0 \dots a_{n-1} \mid n > 1, a_0, \dots, a_{n-1} \in A\}$: the set of all *finite words* over A (free monoid generated by A)
- for $w = a_0 \dots a_{n-1}$ we let $|w| = n$,

Words

- an *alphabet* A is a finite set
- $A^* = \{\varepsilon\} \cup \{a_0 \dots a_{n-1} \mid n > 1, a_0, \dots, a_{n-1} \in A\}$: the set of all *finite words* over A (free monoid generated by A)
- for $w = a_0 \dots a_{n-1}$ we let $|w| = n$,
- $dom(w) = \{0, 1, \dots, |w| - 1\}$,

Words

- an *alphabet* A is a finite set
- $A^* = \{\varepsilon\} \cup \{a_0 \ldots a_{n-1} \mid n > 1, a_0, \ldots, a_{n-1} \in A\}$: the set of all *finite words* over A (free monoid generated by A)
- for $w = a_0 \ldots a_{n-1}$ we let $|w| = n$,
- $dom(w) = \{0, 1, \ldots, |w| - 1\}$,
- $A^\omega = \{a_0 a_1 \ldots \mid a_0, a_1, \ldots \in A\}$: the set of all *infinite words* over A

Words

- an *alphabet* A is a finite set
- $A^* = \{\varepsilon\} \cup \{a_0 \dots a_{n-1} \mid n > 1, a_0, \dots, a_{n-1} \in A\}$: the set of all *finite words* over A (free monoid generated by A)
- for $w = a_0 \dots a_{n-1}$ we let $|w| = n$,
- $dom(w) = \{0, 1, \dots, |w| - 1\}$,
- $A^\omega = \{a_0 a_1 \dots \mid a_0, a_1, \dots \in A\}$: the set of all *infinite words* over A
- for $w = a_0 a_1 \dots$

Words

- an *alphabet* A is a finite set
- $A^* = \{\varepsilon\} \cup \{a_0 \ldots a_{n-1} \mid n > 1, a_0, \ldots, a_{n-1} \in A\}$: the set of all *finite words* over A (free monoid generated by A)
- for $w = a_0 \ldots a_{n-1}$ we let $|w| = n$,
- $dom(w) = \{0, 1, \ldots, |w| - 1\}$,
- $A^\omega = \{a_0 a_1 \ldots \mid a_0, a_1, \ldots \in A\}$: the set of all *infinite words* over A
- for $w = a_0 a_1 \ldots$
- $dom(w) = \omega (= \mathbb{N})$,

Words

- an *alphabet* A is a finite set
- $A^* = \{\varepsilon\} \cup \{a_0 \dots a_{n-1} \mid n > 1, a_0, \dots, a_{n-1} \in A\}$: the set of all *finite words* over A (free monoid generated by A)
- for $w = a_0 \dots a_{n-1}$ we let $|w| = n$,
- $dom(w) = \{0, 1, \dots, |w| - 1\}$,
- $A^\omega = \{a_0 a_1 \dots \mid a_0, a_1, \dots \in A\}$: the set of all *infinite words* over A
- for $w = a_0 a_1 \dots$
- $dom(w) = \omega (= \mathbb{N})$,
- for $w \in A^* \cup A^\omega$, we let $w(i) = a_i$ for every $i \in dom(w)$

Finite automata

- *A finite automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

Finite automata

- A *finite automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*

Finite automata

- A *finite automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*

Finite automata

- A *finite automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*

Finite automata

- A *finite automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*
  - $\Delta \subseteq Q \times A \times Q$: the *set of transitions*

Finite automata

- A *finite automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*
  - $\Delta \subseteq Q \times A \times Q$: the *set of transitions*
  - $F \subseteq Q$: the *final state set*

Finite automata

- A *finite automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*
  - $\Delta \subseteq Q \times A \times Q$: the *set of transitions*
  - $F \subseteq Q$: the *final state set*

- $w = a_0 \ldots a_{n-1} \in A^*$

Finite automata

- A *finite automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*
  - $\Delta \subseteq Q \times A \times Q$: the *set of transitions*
  - $F \subseteq Q$: the *final state set*

- $w = a_0 \ldots a_{n-1} \in A^*$
- a *path of $\mathcal{A}$ over w*

$$P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots (q_{n-1}, a_{n-1}, q_n) \in \Delta^*$$

Finite automata

- A *finite automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*
  - $\Delta \subseteq Q \times A \times Q$: the *set of transitions*
  - $F \subseteq Q$: the *final state set*

- $w = a_0 \dots a_{n-1} \in A^*$
- a *path of* $\mathcal{A}$ *over* w

$$P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \dots (q_{n-1}, a_{n-1}, q_n) \in \Delta^*$$

- P_w: *successful* if $q_0 \in I$ and $q_n \in F$

Finite automata

- A *finite automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*
  - $\Delta \subseteq Q \times A \times Q$: the *set of transitions*
  - $F \subseteq Q$: the *final state set*

- $w = a_0 \ldots a_{n-1} \in A^*$
- a *path of $\mathcal{A}$ over w*

$$P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots (q_{n-1}, a_{n-1}, q_n) \in \Delta^*$$

- P_w: *successful* if $q_0 \in I$ and $q_n \in F$
- $w \in A^*$ is *accepted* (or *recognized*) *by $\mathcal{A}$* if there is a successful path P_w of $\mathcal{A}$ over w

Finite automata

- A *finite automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*
  - $\Delta \subseteq Q \times A \times Q$: the *set of transitions*
  - $F \subseteq Q$: the *final state set*

- $w = a_0 \ldots a_{n-1} \in A^*$
- a *path of $\mathcal{A}$ over w*

$$P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots (q_{n-1}, a_{n-1}, q_n) \in \Delta^*$$

- P_w: *successful* if $q_0 \in I$ and $q_n \in F$
- $w \in A^*$ is *accepted* (or *recognized*) *by* $\mathcal{A}$ if there is a successful path P_w of $\mathcal{A}$ over w
- $L(\mathcal{A})$: the *language of* (all words accepted by) $\mathcal{A}$

Recognizable languages

- $L \subseteq A^*$ is *recognizable* if there is an $\mathcal{A} = (Q, A, I, \Delta, F)$ such that $L = L(\mathcal{A})$

Recognizable languages

- $L \subseteq A^*$ is *recognizable* if there is an $\mathcal{A} = (Q, A, I, \Delta, F)$ such that $L = L(\mathcal{A})$
- $Rec(A)$: the class of all recognizable languages over A

Büchi automata

- A (nondeterministic) *Büchi automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

Büchi automata

- A (nondeterministic) *Büchi automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*

Büchi automata

- A (nondeterministic) *Büchi automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*

Büchi automata

- A (nondeterministic) *Büchi automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*

Büchi automata

- A (nondeterministic) *Büchi automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*
  - $\Delta \subseteq Q \times A \times Q$: the *set of transitions*

Büchi automata

- A (nondeterministic) *Büchi automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*
  - $\Delta \subseteq Q \times A \times Q$: the *set of transitions*
  - $F \subseteq Q$: the *final state set*

Büchi automata

- A (nondeterministic) *Büchi automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*
  - $\Delta \subseteq Q \times A \times Q$: the *set of transitions*
  - $F \subseteq Q$: the *final state set*

- $w = a_0 a_1 \ldots \in A^\omega$

Büchi automata

- A (nondeterministic) *Büchi automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*
  - $\Delta \subseteq Q \times A \times Q$: the *set of transitions*
  - $F \subseteq Q$: the *final state set*
- $w = a_0 a_1 \ldots \in A^\omega$
- a *path of $\mathcal{A}$ over w*

$$P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots \in \Delta^\omega$$

Büchi automata

- A (nondeterministic) *Büchi automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*
  - $\Delta \subseteq Q \times A \times Q$: the *set of transitions*
  - $F \subseteq Q$: the *final state set*

- $w = a_0 a_1 \ldots \in A^\omega$
- a *path of $\mathcal{A}$ over w*

$$P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots \in \Delta^\omega$$

- $In^Q(P_w) = \{q \in Q \mid \exists^\omega i : t_i = (q, a_i, q_{i+1})\}$

Büchi automata

- A (nondeterministic) *Büchi automaton*

$$\mathcal{A} = (Q, A, I, \Delta, F)$$

  - Q: the *finite state set*
  - A: the *input alphabet*
  - $I \subseteq Q$: the *initial state set*
  - $\Delta \subseteq Q \times A \times Q$: the *set of transitions*
  - $F \subseteq Q$: the *final state set*

- $w = a_0 a_1 \ldots \in A^\omega$
- a *path of $\mathcal{A}$ over w*

$$P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots \in \Delta^\omega$$

- $In^Q(P_w) = \{q \in Q \mid \exists^\omega i : t_i = (q, a_i, q_{i+1})\}$
- P_w: *successful* if $q_0 \in I$ and $In^Q(P_w) \cap F \neq \emptyset$

Infinitary recognizable languages

- *$w \in A^{\omega}$ is accepted (or recognized) by $\mathcal{A}$* if there is a successful path P_w of $\mathcal{A}$ over w

Infinitary recognizable languages

- $w \in A^\omega$ is *accepted* (or *recognized*) *by* $\mathcal{A}$ if there is a successful path P_w of $\mathcal{A}$ over w
- $L(\mathcal{A})$: the *language of* (all infinite words accepted by) $\mathcal{A}$

Infinitary recognizable languages

- $w \in A^{\omega}$ is *accepted* (or *recognized*) *by* $\mathcal{A}$ if there is a successful path P_w of $\mathcal{A}$ over w
- $L(\mathcal{A})$: the *language of* (all infinite words accepted by) $\mathcal{A}$
- $L \subseteq A^{\omega}$ is ω-*recognizable* if there is an $\mathcal{A} = (Q, A, I, \Delta, F)$ such that $L = L(\mathcal{A})$

Infinitary recognizable languages

- $w \in A^{\omega}$ is *accepted* (or *recognized*) *by* $\mathcal{A}$ if there is a successful path P_w of $\mathcal{A}$ over w
- $L(\mathcal{A})$: the *language of* (all infinite words accepted by) $\mathcal{A}$
- $L \subseteq A^{\omega}$ is ω-*recognizable* if there is an $\mathcal{A} = (Q, A, I, \Delta, F)$ such that $L = L(\mathcal{A})$
- ω-$Rec(A)$: the class of all ω-recognizable languages over A

MSO logic - Syntax

Definition

The syntax of the MSO-formulas over A is given by

$$\varphi ::= \mathit{true} \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

where $a \in A$, x, y are first-order variables, and X is a second-order variable.

MSO logic - Syntax

Definition

The syntax of the MSO-formulas over A is given by

$$\varphi ::= true \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \centerdot \varphi \mid \exists X \centerdot \varphi$$

where $a \in A$, x, y are first-order variables, and X is a second-order variable.

- $\neg true = false$

MSO logic - Syntax

Definition

The syntax of the MSO-formulas over A is given by

$$\varphi ::= true \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \centerdot \varphi \mid \exists X \centerdot \varphi$$

where $a \in A$, x, y are first-order variables, and X is a second-order variable.

- $\neg true = false$
- $\neg\neg\varphi = \varphi$

MSO logic - Syntax

Definition

The syntax of the MSO-formulas over A is given by

$$\varphi ::= true \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

where $a \in A$, x, y are first-order variables, and X is a second-order variable.

- $\neg true = false$
- $\neg\neg\varphi = \varphi$
- $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$

MSO logic - Syntax

Definition

The syntax of the MSO-formulas over A is given by

$$\varphi ::= true \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\,.\,\varphi \mid \exists X\,.\,\varphi$$

where $a \in A$, x, y are first-order variables, and X is a second-order variable.

- $\neg true = false$
- $\neg\neg\varphi = \varphi$
- $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$
- $\forall x\,.\,\varphi = \neg(\exists x\,.\,\neg\varphi)$

MSO logic - Syntax

Definition

The syntax of the MSO-formulas over A is given by

$$\varphi ::= \mathit{true} \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\,.\,\varphi \mid \exists X\,.\,\varphi$$

where $a \in A$, x, y are first-order variables, and X is a second-order variable.

- $\neg \mathit{true} = \mathit{false}$
- $\neg\neg\varphi = \varphi$
- $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$
- $\forall x\,.\,\varphi = \neg(\exists x\,.\,\neg\varphi)$
- $\forall X\,.\,\varphi = \neg(\exists X\,.\,\neg\varphi)$

MSO logic - Syntax

Definition

The syntax of the MSO-formulas over A is given by

$$\varphi ::= true \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

where $a \in A$, x, y are first-order variables, and X is a second-order variable.

- $\neg true = false$
- $\neg\neg\varphi = \varphi$
- $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$
- $\forall x \,.\, \varphi = \neg(\exists x \,.\, \neg\varphi)$
- $\forall X \,.\, \varphi = \neg(\exists X \,.\, \neg\varphi)$
- $MSO(A)$: the set of all MSO-formulas over A

MSO logic - Syntax

Definition

The syntax of the MSO-formulas over A is given by

$$\varphi ::= true \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

where $a \in A$, x, y are first-order variables, and X is a second-order variable.

- $\neg true = false$
- $\neg\neg\varphi = \varphi$
- $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$
- $\forall x \,.\, \varphi = \neg(\exists x \,.\, \neg\varphi)$
- $\forall X \,.\, \varphi = \neg(\exists X \,.\, \neg\varphi)$
- $MSO(A)$: the set of all MSO-formulas over A
- Example: $\varphi = \exists x \,.\, (\forall y \,.\, (x \leq y) \wedge P_a(x))$

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$ and $w \in A^*$

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$ and $w \in A^*$
- First-order variables in φ represent *positions* in w and second-order variables in φ represent *set of positions* in w

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$ and $w \in A^*$
- First-order variables in φ represent *positions* in w and second-order variables in φ represent *set of positions* in w
- in this way we shall check if w "satisfies" φ

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$ and $w \in A^*$
- First-order variables in φ represent *positions* in w and second-order variables in φ represent *set of positions* in w
- in this way we shall check if w "satisfies" φ
- for instance $\varphi = P_a(x)$ will be satisfied by w if the letter of w at the position represented by x is a

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$ and $w \in A^*$
- First-order variables in φ represent *positions* in w and second-order variables in φ represent *set of positions* in w
- in this way we shall check if w "satisfies" φ
- for instance $\varphi = P_a(x)$ will be satisfied by w if the letter of w at the position represented by x is a
- but *which position is represented by* x?

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$ and $w \in A^*$
- First-order variables in φ represent *positions* in w and second-order variables in φ represent *set of positions* in w
- in this way we shall check if w "satisfies" φ
- for instance $\varphi = P_a(x)$ will be satisfied by w if the letter of w at the position represented by x is a
- but *which position is represented by* x?
- A first- or a second-order variable is called *free* it is not in the scope of any quantifier

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$ and $w \in A^*$
- First-order variables in φ represent *positions* in w and second-order variables in φ represent *set of positions* in w
- in this way we shall check if w "satisfies" φ
- for instance $\varphi = P_a(x)$ will be satisfied by w if the letter of w at the position represented by x is a
- but *which position is represented by* x?
- A first- or a second-order variable is called *free* it is not in the scope of any quantifier
- Example: $\varphi = \forall y \,.\, (x \leq y)$ x is a free variable in φ but not in $\varphi' = \exists x \,.\, \varphi$

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$ and $w \in A^*$
- First-order variables in φ represent *positions* in w and second-order variables in φ represent *set of positions* in w
- in this way we shall check if w "satisfies" φ
- for instance $\varphi = P_a(x)$ will be satisfied by w if the letter of w at the position represented by x is a
- but *which position is represented by* x?
- A first- or a second-order variable is called *free* it is not in the scope of any quantifier
- Example: $\varphi = \forall y \, . \, (x \leq y)$ x is a free variable in φ but not in $\varphi' = \exists x \, . \, \varphi$
- $Free(\varphi)$: the set of free variables of φ

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$ and $w \in A^*$
- First-order variables in φ represent *positions* in w and second-order variables in φ represent *set of positions* in w
- in this way we shall check if w "satisfies" φ
- for instance $\varphi = P_a(x)$ will be satisfied by w if the letter of w at the position represented by x is a
- but *which position is represented by* x?
- A first- or a second-order variable is called *free* it is not in the scope of any quantifier
- Example: $\varphi = \forall y \,.\, (x \leq y)$ x is a free variable in φ but not in $\varphi' = \exists x \,.\, \varphi$
- $Free(\varphi)$: the set of free variables of φ
- In order to define the **semantics** of an MSO-formula φ we have to assign "truth values" to its free variables

MSO logic - Semantics (over finite words)

- $\varphi \in MSO(A)$, $w \in A^*$, $dom(w) = \{0, 1, \ldots, |w| - 1\}$

MSO logic - Semantics (over finite words)

- $\varphi \in MSO(A)$, $w \in A^*$, $dom(w) = \{0, 1, \ldots, |w| - 1\}$
- A $(w, Free(\varphi))$-*assignment* σ is a mapping associating first order variables from $Free(\varphi)$ to elements of $dom(w)$, and second order variables from $Free(\varphi)$ to subsets of $dom(w)$

MSO logic - Semantics (over finite words)

- $\varphi \in MSO(A)$, $w \in A^*$, $dom(w) = \{0, 1, \ldots, |w| - 1\}$
- A $(w, Free(\varphi))$-*assignment* σ is a mapping associating first order variables from $Free(\varphi)$ to elements of $dom(w)$, and second order variables from $Free(\varphi)$ to subsets of $dom(w)$
- if x is a first order variable and $i \in dom(w)$, then $\sigma[x \to i]$ denotes the $(w, Free(\varphi) \cup \{x\})$-assignment which associates i to x and acts as σ on $Free(\varphi) \setminus \{x\}$

MSO logic - Semantics (over finite words)

- $\varphi \in MSO(A)$, $w \in A^*$, $dom(w) = \{0, 1, \ldots, |w| - 1\}$
- A $(w, Free(\varphi))$-*assignment* σ is a mapping associating first order variables from $Free(\varphi)$ to elements of $dom(w)$, and second order variables from $Free(\varphi)$ to subsets of $dom(w)$
- if x is a first order variable and $i \in dom(w)$, then $\sigma[x \to i]$ denotes the $(w, Free(\varphi) \cup \{x\})$-assignment which associates i to x and acts as σ on $Free(\varphi) \setminus \{x\}$
- if X is a second order variable and $I \subseteq dom(w)$, then $\sigma[X \to I]$ denotes the $(w, Free(\varphi) \cup \{X\})$-assignment which associates I to X and acts as σ on $Free(\varphi) \setminus \{X\}$

MSO logic - Semantics (over finite words)

- we use the extended alphabet $A_{Free(\varphi)} = A \times \{0,1\}^{Free(\varphi)}$

MSO logic - Semantics (over finite words)

- we use the extended alphabet $A_{Free(\varphi)} = A \times \{0,1\}^{Free(\varphi)}$
- Example: $w = abbab$ $(dom(w) = \{0,1,2,3,4\})$,
  $Free(\varphi) = \{x, y, X\}$

MSO logic - Semantics (over finite words)

- we use the extended alphabet $A_{Free(\varphi)} = A \times \{0,1\}^{Free(\varphi)}$
- Example: $w = abbab$ $(dom(w) = \{0,1,2,3,4\})$, $Free(\varphi) = \{x, y, X\}$
- σ be a $(w, Free(\varphi))$-assignment with $\sigma(x) = 1, \sigma(y) = 3, \sigma(X) = \{1,2,4\}$

MSO logic - Semantics (over finite words)

- we use the extended alphabet $A_{Free(\varphi)} = A \times \{0,1\}^{Free(\varphi)}$
- Example: $w = abbab$ ($dom(w) = \{0,1,2,3,4\}$), $Free(\varphi) = \{x, y, X\}$
- σ be a $(w, Free(\varphi))$-assignment with $\sigma(x) = 1, \sigma(y) = 3, \sigma(X) = \{1,2,4\}$
- we can represent the word $(w, \sigma) \in A^*_{Free(\varphi)}$ as follows:

MSO logic - Semantics (over finite words)

- we use the extended alphabet $A_{Free(\varphi)} = A \times \{0,1\}^{Free(\varphi)}$
- Example: $w = abbab$ ($dom(w) = \{0,1,2,3,4\}$), $Free(\varphi) = \{x, y, X\}$
- σ be a $(w, Free(\varphi))$-assignment with $\sigma(x) = 1, \sigma(y) = 3, \sigma(X) = \{1,2,4\}$
- we can represent the word $(w, \sigma) \in A^*_{Free(\varphi)}$ as follows:

-

	a	b	b	a	b
x	0	1	0	0	0
y	0	0	0	1	0
X	0	1	1	0	1

MSO logic - Semantics (over finite words)

- Example: $\varphi = P_a(x) \wedge P_b(y)$, $Free(\varphi) = \{x, y\}$

MSO logic - Semantics (over finite words)

- Example: $\varphi = P_a(x) \wedge P_b(y)$, $Free(\varphi) = \{x, y\}$
- $w = abbab$,

MSO logic - Semantics (over finite words)

- Example: $\varphi = P_a(x) \wedge P_b(y)$, $Free(\varphi) = \{x, y\}$
- $w = abbab$,
- (w, σ) by

	a	b	b	a	b
x	0	1	0	0	0
y	0	0	0	0	1

MSO logic - Semantics (over finite words)

- Example: $\varphi = P_a(x) \wedge P_b(y)$, $Free(\varphi) = \{x, y\}$
- $w = abbab$,
- (w, σ) by

	a	b	b	a	b
x	0	1	0	0	0
y	0	0	0	0	1

- $(w, \sigma) \not\models \varphi$

MSO logic - Semantics (over finite words)

- Example: $\varphi = P_a(x) \wedge P_b(y)$, $Free(\varphi) = \{x, y\}$
- $w = abbab$,
- (w, σ) by

	a	b	b	a	b
x	0	1	0	0	0
y	0	0	0	0	1

- $(w, \sigma) \not\models \varphi$
- (w, σ') by

	a	b	b	a	b
x	1	0	0	0	0
y	0	0	1	0	0

MSO logic - Semantics (over finite words)

- Example: $\varphi = P_a(x) \wedge P_b(y)$, $Free(\varphi) = \{x, y\}$
- $w = abbab$,
- (w, σ) by

	a	b	b	a	b
x	0	1	0	0	0
y	0	0	0	0	1

- $(w, \sigma) \not\models \varphi$
- (w, σ') by

	a	b	b	a	b
x	1	0	0	0	0
y	0	0	1	0	0

- $(w, \sigma') \models \varphi$

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$, $w \in A^*$, and σ a $(w, Free(\varphi))$-assignment

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$, $w \in A^*$, and σ a $(w, Free(\varphi))$-assignment
- We define the satisfaction $(w, \sigma) \models \varphi$ of φ by (w, σ) by induction on the structure of φ:

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$, $w \in A^*$, and σ a $(w, Free(\varphi))$-assignment
- We define the satisfaction $(w, \sigma) \models \varphi$ of φ by (w, σ) by induction on the structure of φ:
  - $(w, \sigma) \models true$

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$, $w \in A^*$, and σ a $(w, Free(\varphi))$-assignment
- We define the satisfaction $(w, \sigma) \models \varphi$ of φ by (w, σ) by induction on the structure of φ:
    - $(w, \sigma) \models true$
    - $(w, \sigma) \models P_a(x)$ iff $w(\sigma(x)) = a$

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$, $w \in A^*$, and σ a $(w, Free(\varphi))$-assignment
- We define the satisfaction $(w, \sigma) \models \varphi$ of φ by (w, σ) by induction on the structure of φ:
  - $(w, \sigma) \models true$
  - $(w, \sigma) \models P_a(x)$ iff $w(\sigma(x)) = a$
  - $(w, \sigma) \models x \in X$ iff $\sigma(x) \in \sigma(X)$

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$, $w \in A^*$, and σ a $(w, Free(\varphi))$-assignment
- We define the satisfaction $(w, \sigma) \models \varphi$ of φ by (w, σ) by induction on the structure of φ:
  - $(w, \sigma) \models true$
  - $(w, \sigma) \models P_a(x)$ iff $w(\sigma(x)) = a$
  - $(w, \sigma) \models x \in X$ iff $\sigma(x) \in \sigma(X)$
  - $(w, \sigma) \models x \leq y$ iff $\sigma(x) \leq \sigma(y)$

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$, $w \in A^*$, and σ a $(w, Free(\varphi))$-assignment
- We define the satisfaction $(w, \sigma) \models \varphi$ of φ by (w, σ) by induction on the structure of φ:
  - $(w, \sigma) \models true$
  - $(w, \sigma) \models P_a(x)$ iff $w(\sigma(x)) = a$
  - $(w, \sigma) \models x \in X$ iff $\sigma(x) \in \sigma(X)$
  - $(w, \sigma) \models x \leq y$ iff $\sigma(x) \leq \sigma(y)$
  - $(w, \sigma) \models \neg\varphi$ iff $(w, \sigma) \not\models \varphi$

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$, $w \in A^*$, and σ a $(w, Free(\varphi))$-assignment
- We define the satisfaction $(w, \sigma) \models \varphi$ of φ by (w, σ) by induction on the structure of φ:
  - $(w, \sigma) \models true$
  - $(w, \sigma) \models P_a(x)$ iff $w(\sigma(x)) = a$
  - $(w, \sigma) \models x \in X$ iff $\sigma(x) \in \sigma(X)$
  - $(w, \sigma) \models x \leq y$ iff $\sigma(x) \leq \sigma(y)$
  - $(w, \sigma) \models \neg\varphi$ iff $(w, \sigma) \not\models \varphi$
  - $(w, \sigma) \models \varphi \vee \psi$ iff $(w, \sigma) \models \varphi$ or $(w, \sigma) \models \psi$

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$, $w \in A^*$, and σ a $(w, Free(\varphi))$-assignment
- We define the satisfaction $(w, \sigma) \models \varphi$ of φ by (w, σ) by induction on the structure of φ:
  - $(w, \sigma) \models true$
  - $(w, \sigma) \models P_a(x)$ iff $w(\sigma(x)) = a$
  - $(w, \sigma) \models x \in X$ iff $\sigma(x) \in \sigma(X)$
  - $(w, \sigma) \models x \leq y$ iff $\sigma(x) \leq \sigma(y)$
  - $(w, \sigma) \models \neg\varphi$ iff $(w, \sigma) \not\models \varphi$
  - $(w, \sigma) \models \varphi \vee \psi$ iff $(w, \sigma) \models \varphi$ or $(w, \sigma) \models \psi$
  - $(w, \sigma) \models \exists x . \varphi$ iff there exists an $i \in dom(w)$ such that $(w, \sigma[x \to i]) \models \varphi$

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$, $w \in A^*$, and σ a $(w, Free(\varphi))$-assignment
- We define the satisfaction $(w, \sigma) \models \varphi$ of φ by (w, σ) by induction on the structure of φ:
  - $(w, \sigma) \models true$
  - $(w, \sigma) \models P_a(x)$ iff $w(\sigma(x)) = a$
  - $(w, \sigma) \models x \in X$ iff $\sigma(x) \in \sigma(X)$
  - $(w, \sigma) \models x \leq y$ iff $\sigma(x) \leq \sigma(y)$
  - $(w, \sigma) \models \neg\varphi$ iff $(w, \sigma) \not\models \varphi$
  - $(w, \sigma) \models \varphi \vee \psi$ iff $(w, \sigma) \models \varphi$ or $(w, \sigma) \models \psi$
  - $(w, \sigma) \models \exists x . \varphi$ iff there exists an $i \in dom(w)$ such that $(w, \sigma[x \rightarrow i]) \models \varphi$
  - $(w, \sigma) \models \exists X . \varphi$ iff there exists an $I \subseteq dom(w)$ such that $(w, \sigma[X \rightarrow I]) \models \varphi$

MSO logic - Semantics (over finite words)

- Let $\varphi \in MSO(A)$, $w \in A^*$, and σ a $(w, Free(\varphi))$-assignment
- We define the satisfaction $(w,\sigma) \models \varphi$ of φ by (w,σ) by induction on the structure of φ:
  - $(w,\sigma) \models true$
  - $(w,\sigma) \models P_a(x)$ iff $w(\sigma(x)) = a$
  - $(w,\sigma) \models x \in X$ iff $\sigma(x) \in \sigma(X)$
  - $(w,\sigma) \models x \leq y$ iff $\sigma(x) \leq \sigma(y)$
  - $(w,\sigma) \models \neg\varphi$ iff $(w,\sigma) \not\models \varphi$
  - $(w,\sigma) \models \varphi \vee \psi$ iff $(w,\sigma) \models \varphi$ or $(w,\sigma) \models \psi$
  - $(w,\sigma) \models \exists x \,.\, \varphi$ iff there exists an $i \in dom(w)$ such that $(w,\sigma[x \to i]) \models \varphi$
  - $(w,\sigma) \models \exists X \,.\, \varphi$ iff there exists an $I \subseteq dom(w)$ such that $(w,\sigma[X \to I]) \models \varphi$
- $L(\varphi) = \{(w,\sigma) \in A^*_{Free(\varphi)} \mid (w,\sigma) \models \varphi\}$ the *language of* (all words satisfying) φ

MSO logic - Semantics (over finite words)

- $\varphi \in MSO(A)$ is a *sentence* if $Free(\varphi) = \emptyset$

MSO logic - Semantics (over finite words)

- $\varphi \in MSO(A)$ is a *sentence* if $Free(\varphi) = \emptyset$
- if φ is a sentence, then $L(\varphi) \subseteq A^*$

MSO logic - Semantics (over finite words)

- $\varphi \in MSO(A)$ is a *sentence* if $Free(\varphi) = \emptyset$
- if φ is a sentence, then $L(\varphi) \subseteq A^*$
- Example: Let $\varphi = \exists x . (\forall y . (x \leq y) \wedge P_b(x))$, then $L(\varphi) = bA^*$

MSO logic - Semantics (over finite words)

- $\varphi \in MSO(A)$ is a *sentence* if $Free(\varphi) = \emptyset$
- if φ is a sentence, then $L(\varphi) \subseteq A^*$
- Example: Let $\varphi = \exists x \,.\, (\forall y \,.\, (x \leq y) \wedge P_b(x))$, then $L(\varphi) = bA^*$
- $L \subseteq A^*$ is *MSO-definable* if there is a sentence $\varphi \in MSO(A)$ such that $L = L(\varphi)$

MSO logic - Semantics (over finite words)

- $\varphi \in MSO(A)$ is a *sentence* if $Free(\varphi) = \emptyset$
- if φ is a sentence, then $L(\varphi) \subseteq A^*$
- Example: Let $\varphi = \exists x \centerdot (\forall y \centerdot (x \leq y) \wedge P_b(x))$, then $L(\varphi) = bA^*$
- $L \subseteq A^*$ is *MSO-definable* if there is a sentence $\varphi \in MSO(A)$ such that $L = L(\varphi)$
- $Mso(A)$: the class of all MSO-definable languages over A

MSO logic - Semantics (over finite words)

- $\varphi \in MSO(A)$ is a *sentence* if $Free(\varphi) = \emptyset$
- if φ is a sentence, then $L(\varphi) \subseteq A^*$
- Example: Let $\varphi = \exists x \,.\, (\forall y \,.\, (x \leq y) \wedge P_b(x))$, then $L(\varphi) = bA^*$
- $L \subseteq A^*$ is *MSO-definable* if there is a sentence $\varphi \in MSO(A)$ such that $L = L(\varphi)$
- $Mso(A)$: the class of all MSO-definable languages over A
- J.R. Büchi 1960, C. Elgot 1961, B. Trakhtenbrot 1961:

MSO logic - Semantics (over finite words)

- $\varphi \in MSO(A)$ is a *sentence* if $Free(\varphi) = \emptyset$
- if φ is a sentence, then $L(\varphi) \subseteq A^*$
- Example: Let $\varphi = \exists x \,.\, (\forall y \,.\, (x \leq y) \wedge P_b(x))$, then $L(\varphi) = bA^*$
- $L \subseteq A^*$ is *MSO-definable* if there is a sentence $\varphi \in MSO(A)$ such that $L = L(\varphi)$
- $Mso(A)$: the class of all MSO-definable languages over A
- J.R. Büchi 1960, C. Elgot 1961, B. Trakhtenbrot 1961:
- $$Rec(A) = Mso(A)$$

MSO logic - Semantics (over infinite words)

- Let $\varphi \in MSO(A)$, $w \in A^\omega$, and σ a $(w, Free(\varphi))$-assignment
- We define the satisfaction $(w, \sigma) \models \varphi$ of φ by (w, σ) by induction on the structure of φ:
    - $(w, \sigma) \models P_a(x)$ iff $w(\sigma(x)) = a$
    - $(w, \sigma) \models x \in X$ iff $\sigma(x) \in \sigma(X)$
    - $(w, \sigma) \models x \leq y$ iff $\sigma(x) \leq \sigma(y)$
    - $(w, \sigma) \models \neg\varphi$ iff $(w, \sigma) \not\models \varphi$
    - $(w, \sigma) \models \varphi \vee \psi$ iff $(w, \sigma) \models \varphi$ or $(w, \sigma) \models \psi$
    - $(w, \sigma) \models \exists x . \varphi$ iff there exists an $i \geq 0$ such that $(w, \sigma[x \to i]) \models \varphi$
    - $(w, \sigma) \models \exists X . \varphi$ iff there exists an $I \subseteq \omega$ such that $(w, \sigma[X \to I]) \models \varphi$
- $L(\varphi) = \{(w, \sigma) \in A^\omega_{Free(\varphi)} \mid (w, \sigma) \models \varphi\}$ the *language of* (all infinite words satisfying) φ

MSO logic - Semantics (over infinite words)

- if φ is a sentence, then $L(\varphi) \subseteq A^{\omega}$

MSO logic - Semantics (over infinite words)

- if φ is a sentence, then $L(\varphi) \subseteq A^\omega$
- Example: Let $\varphi = \exists x . (\forall y . (x \leq y) \wedge P_b(x))$, then $L(\varphi) = bA^\omega$

MSO logic - Semantics (over infinite words)

- if φ is a sentence, then $L(\varphi) \subseteq A^\omega$
- Example: Let $\varphi = \exists x \,.\, (\forall y \,.\, (x \leq y) \wedge P_b(x))$, then $L(\varphi) = bA^\omega$
- $L \subseteq A^\omega$ is *MSO-definable* if there is a sentence $\varphi \in MSO(A)$ such that $L = L(\varphi)$

MSO logic - Semantics (over infinite words)

- if φ is a sentence, then $L(\varphi) \subseteq A^\omega$
- Example: Let $\varphi = \exists x \,.\, (\forall y \,.\, (x \leq y) \wedge P_b(x))$, then $L(\varphi) = bA^\omega$
- $L \subseteq A^\omega$ is *MSO-definable* if there is a sentence $\varphi \in MSO(A)$ such that $L = L(\varphi)$
- *ω-Mso(A)*: the class of all infinitary MSO-definable languages over A

MSO logic - Semantics (over infinite words)

- if φ is a sentence, then $L(\varphi) \subseteq A^\omega$
- Example: Let $\varphi = \exists x \,.\, (\forall y \,.\, (x \leq y) \wedge P_b(x))$, then $L(\varphi) = bA^\omega$
- $L \subseteq A^\omega$ is *MSO-definable* if there is a sentence $\varphi \in MSO(A)$ such that $L = L(\varphi)$
- ω-$Mso(A)$: the class of all infinitary MSO-definable languages over A
- J. R. Büchi 1962:

MSO logic - Semantics (over infinite words)

- if φ is a sentence, then $L(\varphi) \subseteq A^{\omega}$
- Example: Let $\varphi = \exists x \,.\, (\forall y \,.\, (x \leq y) \wedge P_b(x))$, then $L(\varphi) = bA^{\omega}$
- $L \subseteq A^{\omega}$ is *MSO-definable* if there is a sentence $\varphi \in MSO(A)$ such that $L = L(\varphi)$
- $\omega\text{-}Mso(A)$: the class of all infinitary MSO-definable languages over A
- J. R. Büchi 1962:
-
$$\omega\text{-}Rec(A) = \omega\text{-}Mso(A)$$

Semirings

- $(K, +, \cdot, 0, 1)$: semiring (simply denoted by K) where
  - $+$ is a binary associative and commutative operation on K with neutral element 0, i.e.,
    - $k + (l + m) = (k + l) + m$,
    - $k + l = l + k$,
    - $k + 0 = k$,

    for every $k, l, m \in K$
  - $\cdot$ is a binary associative operation on K with neutral element 1,
    - $k \cdot (l \cdot m) = (k \cdot l) \cdot m$,
    - $k \cdot 1 = 1 \cdot k = 1$,
  - $\cdot$ distributes over $+$, i.e.,
    $k \cdot (l + m) = k \cdot l + k \cdot m$, and
    $(k + l) \cdot m = k \cdot m + l \cdot m$
    for every $k, l, m \in K$, and
  - $k \cdot 0 = 0 \cdot k = 0$ for every $k \in K$.
- if $\cdot$ is commutative, then K is called commutative
- In the sequel: K a commutative semiring

Formal power series

- A finitary formal (power) series over A and K

$$s : A^* \longrightarrow K$$

Formal power series

- A finitary formal (power) series over A and K

$$s : A^* \longrightarrow K$$

- for $w \in A^*$ the value $s(w)$ is called the *coefficient of s at w* and denoted as (s, w)

Formal power series

- A finitary formal (power) series over A and K

$$s : A^* \longrightarrow K$$

- for $w \in A^*$ the value $s(w)$ is called the *coefficient of s at w* and denoted as (s, w)
- some operations on series: let s_1, s_2 series over A and K and $k \in K$

Formal power series

- A finitary formal (power) series over A and K

$$s : A^* \longrightarrow K$$

- for $w \in A^*$ the value $s(w)$ is called the *coefficient of s at w* and denoted as (s, w)
- some operations on series: let s_1, s_2 series over A and K and $k \in K$
  - *sum* $s_1 + s_2$, $\quad (s_1 + s_2, w) = (s_1, w) + (s_2, w)$

Formal power series

- A finitary formal (power) series over A and K

$$s : A^* \longrightarrow K$$

- for $w \in A^*$ the value $s(w)$ is called the *coefficient of s at w* and denoted as (s, w)
- some operations on series: let s_1, s_2 series over A and K and $k \in K$
  - *sum* $s_1 + s_2$, $(s_1 + s_2, w) = (s_1, w) + (s_2, w)$
  - *scalar product* $k \cdot s_1$, $(k \cdot s_1, w) = k \cdot (s_1, w)$

Formal power series

- A finitary formal (power) series over A and K

$$s : A^* \longrightarrow K$$

- for $w \in A^*$ the value $s(w)$ is called the *coefficient of* s *at* w and denoted as (s, w)
- some operations on series: let s_1, s_2 series over A and K and $k \in K$
  - *sum* $s_1 + s_2$, $\quad (s_1 + s_2, w) = (s_1, w) + (s_2, w)$
  - *scalar product* $k \cdot s_1$, $\quad (k \cdot s_1, w) = k \cdot (s_1, w)$
  - *Hadamard product* $s_1 \odot s_2$, $\quad (s_1 \odot s_2, w) = (s_1, w) \cdot (s_2, w)$ for every $w \in A^*$

Weighted automata

- A weighted automaton over K:

$$\mathcal{A} = (Q, A, in, wt, ter)$$

Weighted automata

- A weighted automaton over K:

$$\mathcal{A} = (Q, A, in, wt, ter)$$

  - Q the *finite state set,*

Weighted automata

- A weighted automaton over K:

$$\mathcal{A} = (Q, A, in, wt, ter)$$

  - Q the *finite state set*,
  - A the *input alphabet*,

Weighted automata

- A weighted automaton over K:

$$\mathcal{A} = (Q, A, in, wt, ter)$$

  - Q the *finite state set,*
  - A the *input alphabet,*
  - $in : Q \to K$ the *initial distribution,*

Weighted automata

- A weighted automaton over K:

$$\mathcal{A} = (Q, A, in, wt, ter)$$

  - Q the *finite state set,*
  - A the *input alphabet,*
  - $in : Q \to K$ the *initial distribution,*
  - $wt : Q \times A \times Q \to K$ the *weight assignment mapping,*

Weighted automata

- A weighted automaton over K:

$$\mathcal{A} = (Q, A, in, wt, ter)$$

  - Q the *finite state set,*
  - A the *input alphabet,*
  - $in : Q \to K$ the *initial distribution,*
  - $wt : Q \times A \times Q \to K$ the *weight assignment mapping,*
  - $ter : Q \to K$ the *terminal distribution*

Weighted automata

- A weighted automaton over K:

$$\mathcal{A} = (Q, A, in, wt, ter)$$

  - Q the *finite state set,*
  - A the *input alphabet,*
  - $in : Q \to K$ the *initial distribution,*
  - $wt : Q \times A \times Q \to K$ the *weight assignment mapping,*
  - $ter : Q \to K$ the *terminal distribution*

- $w = a_0 \ldots a_{n-1} \in A^*$

Weighted automata

- A weighted automaton over K:

$$\mathcal{A} = (Q, A, in, wt, ter)$$

  - Q the *finite state set,*
  - A the *input alphabet,*
  - $in : Q \to K$ the *initial distribution,*
  - $wt : Q \times A \times Q \to K$ the *weight assignment mapping,*
  - $ter : Q \to K$ the *terminal distribution*
- $w = a_0 \ldots a_{n-1} \in A^*$
- a *path of $\mathcal{A}$ over w*

$$P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots (q_{n-1}, a_{n-1}, q_n)$$

where $(q_i, a_i, q_{i+1}) \in Q \times A \times Q$ for every $0 \leq i \leq n-1$

Weighted automata

- A weighted automaton over K:

$$\mathcal{A} = (Q, A, in, wt, ter)$$

  - Q the *finite state set,*
  - A the *input alphabet,*
  - $in : Q \to K$ the *initial distribution,*
  - $wt : Q \times A \times Q \to K$ the *weight assignment mapping,*
  - $ter : Q \to K$ the *terminal distribution*
- $w = a_0 \dots a_{n-1} \in A^*$
- a *path of $\mathcal{A}$ over w*

$$P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \dots (q_{n-1}, a_{n-1}, q_n)$$

where $(q_i, a_i, q_{i+1}) \in Q \times A \times Q$ for every $0 \leq i \leq n-1$

- the *weight of P_w*:

$$weight(P_w) = in(q_0) \cdot wt((q_0, a_0, q_1)) \cdot wt((q_1, a_1, q_2)) \cdot \dots \cdot wt((q_{n-1}, a_{n-1}, q_n)) \cdot ter(q_n)$$

Weighted automata

- the *behavior of* $\mathcal{A}$ is the series

$$\|\mathcal{A}\| : A^* \to K$$

defined for every $w \in A^*$ by

$$(\|\mathcal{A}\|, w) = \sum_{P_w} weight(P_w)$$

- Example: A finite automaton $\mathcal{A} = (Q, A, I, \Delta, F)$ can be considered as a weighted automaton $\mathcal{A}' = (Q, A, in, wt, ter)$ over the *Boolean semiring* $(\{0, 1\}, +, \cdot, 0, 1)$, where:

Weighted automata

- Example: A finite automaton $\mathcal{A} = (Q, A, I, \Delta, F)$ can be considered as a weighted automaton $\mathcal{A}' = (Q, A, in, wt, ter)$ over the *Boolean semiring* $(\{0,1\}, +, \cdot, 0, 1)$, where:
- $in(q) = \begin{cases} 1 & \text{if } q \in I \\ 0 & \text{otherwise} \end{cases}$,

Weighted automata

- Example: A finite automaton $\mathcal{A} = (Q, A, I, \Delta, F)$ can be considered as a weighted automaton $\mathcal{A}' = (Q, A, in, wt, ter)$ over the *Boolean semiring* $(\{0,1\}, +, \cdot, 0, 1)$, where:
- $in(q) = \begin{cases} 1 & \text{if } q \in I \\ 0 & \text{otherwise} \end{cases}$,
- $wt((q, a, q')) = \begin{cases} 1 & \text{if } (q, a, q') \in \Delta \\ 0 & \text{otherwise} \end{cases}$, and

- Example: A finite automaton $\mathcal{A} = (Q, A, I, \Delta, F)$ can be considered as a weighted automaton $\mathcal{A}' = (Q, A, in, wt, ter)$ over the *Boolean semiring* $(\{0,1\}, +, \cdot, 0, 1)$, where:
- $in(q) = \begin{cases} 1 & \text{if } q \in I \\ 0 & \text{otherwise} \end{cases}$,
- $wt((q, a, q')) = \begin{cases} 1 & \text{if } (q, a, q') \in \Delta \\ 0 & \text{otherwise} \end{cases}$, and
- $ter(q) = \begin{cases} 1 & \text{if } q \in F \\ 0 & \text{otherwise} \end{cases}$

Weighted automata

- Example: A finite automaton $\mathcal{A} = (Q, A, I, \Delta, F)$ can be considered as a weighted automaton $\mathcal{A}' = (Q, A, in, wt, ter)$ over the *Boolean semiring* $(\{0,1\}, +, \cdot, 0, 1)$, where:
- $in(q) = \begin{cases} 1 & \text{if } q \in I \\ 0 & \text{otherwise} \end{cases}$,
- $wt((q,a,q')) = \begin{cases} 1 & \text{if } (q,a,q') \in \Delta \\ 0 & \text{otherwise} \end{cases}$, and
- $ter(q) = \begin{cases} 1 & \text{if } q \in F \\ 0 & \text{otherwise} \end{cases}$
- Then a word $w \in A^*$ is a accepted by $\mathcal{A}$ iff $(\|\mathcal{A}'\|, w) = 1$

Recognizable series

- A series s over A and K is *recognizable* if there exists a weighted automaton $\mathcal{A}$ over A and K such that $s = \|\mathcal{A}\|$

Recognizable series

- A series s over A and K is *recognizable* if there exists a weighted automaton $\mathcal{A}$ over A and K such that $s = \|\mathcal{A}\|$
- $Rec(A, K)$: the class of all recognizable series over A and K

Semirings with infinite sums and products

- In order to compute the weights of infinite paths as well as the behavior over infinite words, we assume in the sequel that the semiring K permits infinite sums and products

Semirings with infinite sums and products

- In order to compute the weights of infinite paths as well as the behavior over infinite words, we assume in the sequel that the semiring K permits infinite sums and products
- Examples of such semirings:

- In order to compute the weights of infinite paths as well as the behavior over infinite words, we assume in the sequel that the semiring K permits infinite sums and products
- Examples of such semirings:
  - $(\{0,1\},+,\cdot,0,1)$ the *Boolean* semiring,

Semirings with infinite sums and products

- In order to compute the weights of infinite paths as well as the behavior over infinite words, we assume in the sequel that the semiring K permits infinite sums and products
- Examples of such semirings:
  - $(\{0, 1\}, +, \cdot, 0, 1)$ the *Boolean* semiring,
  - $(\mathbb{N} \cup \{\infty\}, +, \cdot, 0, 1)$ the semiring of *extended natural numbers*,

Semirings with infinite sums and products

- In order to compute the weights of infinite paths as well as the behavior over infinite words, we assume in the sequel that the semiring K permits infinite sums and products
- Examples of such semirings:
  - $(\{0,1\},+,\cdot,0,1)$ the *Boolean* semiring,
  - $(\mathbb{N} \cup \{\infty\},+,\cdot,0,1)$ the semiring of *extended natural numbers*,
  - $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ where $\mathbb{R}_+ = \{r \in \mathbb{R} \mid r \geq 0\}$ the *min-plus semiring*,

Semirings with infinite sums and products

- In order to compute the weights of infinite paths as well as the behavior over infinite words, we assume in the sequel that the semiring K permits infinite sums and products
- Examples of such semirings:
  - $(\{0,1\},+,\cdot,0,1)$ the *Boolean* semiring,
  - $(\mathbb{N}\cup\{\infty\},+,\cdot,0,1)$ the semiring of *extended natural numbers*,
  - $(\mathbb{R}_+\cup\{\infty\},\min,+,\infty,0)$ where $\mathbb{R}_+=\{r\in\mathbb{R}\mid r\geq 0\}$ the *min-plus semiring*,
  - $(\mathbb{R}_+\cup\{-\infty,\infty\},\sup,+,-\infty,0)$ the *max-plus semiring with infinity*,

Semirings with infinite sums and products

- In order to compute the weights of infinite paths as well as the behavior over infinite words, we assume in the sequel that the semiring K permits infinite sums and products
- Examples of such semirings:
  - $(\{0,1\},+,\cdot,0,1)$ the *Boolean* semiring,
  - $(\mathbb{N}\cup\{\infty\},+,\cdot,0,1)$ the semiring of *extended natural numbers*,
  - $(\mathbb{R}_+\cup\{\infty\},\min,+,\infty,0)$ where $\mathbb{R}_+=\{r\in\mathbb{R}\mid r\geq 0\}$ the *min-plus semiring*,
  - $(\mathbb{R}_+\cup\{-\infty,\infty\},\sup,+,-\infty,0)$ the *max-plus semiring with infinity*,
  - $F=([0,1],\sup,\inf,0,1)$ the *fuzzy semiring*

- An infinitary formal (power) series over A and K

$$s : A^{\omega} \longrightarrow K$$

Infinitary formal power series

- An infinitary formal (power) series over A and K

$$s : A^\omega \longrightarrow K$$

- for $w \in A^\omega$ the value $s(w)$ is called the *coefficient of s at w* and denoted as (s, w)

Infinitary formal power series

- An infinitary formal (power) series over A and K

$$s : A^{\omega} \longrightarrow K$$

- for $w \in A^{\omega}$ the value $s(w)$ is called the *coefficient of s at w* and denoted as (s, w)
- some operations on infinitary series: let s_1, s_2 infinitary series over A and K and $k \in K$

Infinitary formal power series

- An infinitary formal (power) series over A and K

$$s : A^\omega \longrightarrow K$$

- for $w \in A^\omega$ the value $s(w)$ is called the *coefficient of s at w* and denoted as (s, w)
- some operations on infinitary series: let s_1, s_2 infinitary series over A and K and $k \in K$
  - *sum* $s_1 + s_2$, $(s_1 + s_2, w) = (s_1, w) + (s_2, w)$

Infinitary formal power series

- An infinitary formal (power) series over A and K

$$s : A^{\omega} \longrightarrow K$$

- for $w \in A^{\omega}$ the value $s(w)$ is called the *coefficient of s at w* and denoted as (s, w)
- some operations on infinitary series: let s_1, s_2 infinitary series over A and K and $k \in K$
  - *sum* $s_1 + s_2$, $\quad (s_1 + s_2, w) = (s_1, w) + (s_2, w)$
  - *scalar product* $k \cdot s_1$, $\quad (k \cdot s_1, w) = k \cdot (s_1, w)$

Infinitary formal power series

- An infinitary formal (power) series over A and K

$$s : A^\omega \longrightarrow K$$

- for $w \in A^\omega$ the value $s(w)$ is called the *coefficient of s at w* and denoted as (s, w)
- some operations on infinitary series: let s_1, s_2 infinitary series over A and K and $k \in K$
  - *sum* $s_1 + s_2$, $(s_1 + s_2, w) = (s_1, w) + (s_2, w)$
  - *scalar product* $k \cdot s_1$, $(k \cdot s_1, w) = k \cdot (s_1, w)$
  - *Hadamard product* $s_1 \odot s_2$, $(s_1 \odot s_2, w) = (s_1, w) \cdot (s_2, w)$ for every $w \in A^\omega$

Weighted Büchi automata

- A weighted Büchi automaton over K:

$$\mathcal{A} = (Q, A, in, wt, F)$$

Weighted Büchi automata

- A weighted Büchi automaton over K:

$$\mathcal{A} = (Q, A, in, wt, F)$$

  - Q the *finite state set,*

Weighted Büchi automata

- A weighted Büchi automaton over K:

$$\mathcal{A} = (Q, A, in, wt, F)$$

  - Q the *finite state set*,
  - A the *input alphabet*,

Weighted Büchi automata

- A weighted Büchi automaton over K:

$$\mathcal{A} = (Q, A, in, wt, F)$$

  - Q the *finite state set*,
  - A the *input alphabet*,
  - $in : Q \to K$ the *initial distribution*,

Weighted Büchi automata

- A weighted Büchi automaton over K:

$$\mathcal{A} = (Q, A, in, wt, F)$$

  - Q the *finite state set*,
  - A the *input alphabet*,
  - $in : Q \to K$ the *initial distribution*,
  - $wt : Q \times A \times Q \to K$ the *weight assignment mapping*,

Weighted Büchi automata

- A weighted Büchi automaton over K:

$$\mathcal{A} = (Q, A, in, wt, F)$$

  - Q the *finite state set,*
  - A the *input alphabet,*
  - $in : Q \to K$ the *initial distribution,*
  - $wt : Q \times A \times Q \to K$ the *weight assignment mapping,*
  - F the *final state set*

Weighted Büchi automata

- A weighted Büchi automaton over K:

$$\mathcal{A} = (Q, A, in, wt, F)$$

  - Q the *finite state set*,
  - A the *input alphabet*,
  - $in : Q \to K$ the *initial distribution*,
  - $wt : Q \times A \times Q \to K$ the *weight assignment mapping*,
  - F the *final state set*

- $w = a_0 a_1 \ldots \in A^\omega$

Weighted Büchi automata

- A weighted Büchi automaton over K:

$$\mathcal{A} = (Q, A, in, wt, F)$$

  - Q the *finite state set*,
  - A the *input alphabet*,
  - $in : Q \to K$ the *initial distribution*,
  - $wt : Q \times A \times Q \to K$ the *weight assignment mapping*,
  - F the *final state set*
- $w = a_0 a_1 \ldots \in A^\omega$
- a *path of $\mathcal{A}$ over w*

$$P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots$$

where $(q_i, a_i, q_{i+1}) \in Q \times A \times Q$ for every $i \geq 0$

Weighted Büchi automata

- A weighted Büchi automaton over K:

$$\mathcal{A} = (Q, A, in, wt, F)$$

  - Q the *finite state set*,
  - A the *input alphabet*,
  - $in : Q \to K$ the *initial distribution*,
  - $wt : Q \times A \times Q \to K$ the *weight assignment mapping*,
  - F the *final state set*

- $w = a_0 a_1 \ldots \in A^\omega$
- a *path of $\mathcal{A}$ over w*

$$P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots$$

where $(q_i, a_i, q_{i+1}) \in Q \times A \times Q$ for every $i \geq 0$

- the *weight of P_w*:

$$weight(P_w) = in(q_0) \cdot wt((q_0, a_0, q_1)) \cdot wt((q_1, a_1, q_2)) \cdot \ldots$$

Weighted automata

- P_w: *successful* if $In^Q(P_w) \cap F \neq \emptyset$

Weighted automata

- P_w: *successful* if $In^Q(P_w) \cap F \neq \emptyset$
- observe that a successful path P_w can have $weight(P_w) = 0$

- P_w: *successful* if $In^Q(P_w) \cap F \neq \emptyset$
- observe that a successful path P_w can have $weight(P_w) = 0$
- the *behavior of* $\mathcal{A}$ is the infinitary series

$$\|\mathcal{A}\| : A^\omega \to K$$

defined for every $w \in A^\omega$ by

$$(\|\mathcal{A}\|, w) = \sum_{P_w \text{ successful}} weight(P_w)$$

Infinitary recognizable series

- An infintary series s over A and K is *ω-recognizable* if there exists a weighted Büchi automaton $\mathcal{A}$ over A and K such that $s = \|\mathcal{A}\|$

Infinitary recognizable series

- An infintary series s over A and K is *ω-recognizable* if there exists a weighted Büchi automaton $\mathcal{A}$ over A and K such that $s = \|\mathcal{A}\|$
- ω-*Rec*(A, K): the class of all recognizable series over A and K

Weighted MSO logic

- Recall the syntax of the MSO logic

$$\varphi ::= true \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\,.\,\varphi \mid \exists X\,.\,\varphi$$

Weighted MSO logic

- Recall the syntax of the MSO logic

$$\varphi ::= true \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

- We aim to define a *weighted MSO logic* (*wMSO* for short) over the semiring K, i.e, to replace *true* (and *false*) with any value $k \in K$

$$\varphi ::= k \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

Weighted MSO logic

- Recall the syntax of the MSO logic

$$\varphi ::= true \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

- We aim to define a *weighted MSO logic* (*wMSO* for short) over the semiring K, i.e, to replace *true* (and *false*) with any value $k \in K$

$$\varphi ::= k \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

- Problem:

Weighted MSO logic

- Recall the syntax of the MSO logic

$$\varphi ::= true \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

- We aim to define a *weighted MSO logic* (*wMSO* for short) over the semiring K, i.e, to replace *true* (and *false*) with any value $k \in K$

$$\varphi ::= k \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

- Problem:
  - how can we define $\neg k$ for every $k \in K$?

Weighted MSO logic

- Recall the syntax of the MSO logic

$$\varphi ::= true \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \centerdot \varphi \mid \exists X \centerdot \varphi$$

- We aim to define a *weighted MSO logic* (*wMSO* for short) over the semiring K, i.e, to replace *true* (and *false*) with any value $k \in K$

$$\varphi ::= k \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \centerdot \varphi \mid \exists X \centerdot \varphi$$

- Problem:
  - how can we define $\neg k$ for every $k \in K$?
- Solution: we can set

Weighted MSO logic

- Recall the syntax of the MSO logic

$$\varphi ::= true \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

- We aim to define a *weighted MSO logic* (*wMSO* for short) over the semiring K, i.e, to replace *true* (and *false*) with any value $k \in K$

$$\varphi ::= k \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

- Problem:
  - how can we define $\neg k$ for every $k \in K$?
- Solution: we can set
  - $\neg 0 = 1$ and $\neg k = 0$ for $k \neq 0$

Weighted MSO logic

- Recall the syntax of the MSO logic

$$\varphi ::= true \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

- We aim to define a *weighted MSO logic* (*wMSO* for short) over the semiring K, i.e, to replace *true* (and *false*) with any value $k \in K$

$$\varphi ::= k \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

- Problem:
  - how can we define $\neg k$ for every $k \in K$?
- Solution: we can set
  - $\neg 0 = 1$ and $\neg k = 0$ for $k \neq 0$
  - but then the relations $\neg\neg\varphi = \varphi$, $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$, $\forall x \,.\, \varphi = \neg(\exists x \,.\, \neg\varphi)$ $\forall X \,.\, \varphi = \neg(\exists X \,.\, \neg\varphi)$

Weighted MSO logic

- Recall the syntax of the MSO logic

$$\varphi ::= true \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

- We aim to define a *weighted MSO logic* (*wMSO* for short) over the semiring K, i.e, to replace *true* (and *false*) with any value $k \in K$

$$\varphi ::= k \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi$$

- Problem:
    - how can we define $\neg k$ for every $k \in K$?
- Solution: we can set
    - $\neg 0 = 1$ and $\neg k = 0$ for $k \neq 0$
    - but then the relations
      $\neg\neg\varphi = \varphi, \;\; \varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$,
      $\forall x \,.\, \varphi = \neg(\exists x \,.\, \neg\varphi)$
      $\forall X \,.\, \varphi = \neg(\exists X \,.\, \neg\varphi)$
    - will not hold any more!

Weighted MSO logic - Syntax

Definition

The syntax of the *wMSO-formulas over A and K* is given by

$$\varphi ::= k \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg P_a(x) \mid \neg(x \in X) \mid \neg(x \leq y) \\ \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi \mid \forall x \,.\, \varphi$$

where $a \in A$ and $k \in K$.

Weighted MSO logic - Syntax

Definition

The syntax of the *wMSO-formulas over A and K* is given by

$$\varphi ::= k \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg P_a(x) \mid \neg(x \in X) \mid \neg(x \leq y) \\ \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi \mid \forall x \,.\, \varphi$$

where $a \in A$ and $k \in K$.

- We do not need $\forall X \,.\, \varphi$

Weighted MSO logic - Syntax

Definition

The syntax of the *wMSO-formulas over A and K* is given by

$$\varphi ::= k \mid P_a(x) \mid x \in X \mid x \leq y \mid \neg P_a(x) \mid \neg(x \in X) \mid \neg(x \leq y) \\ \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x \,.\, \varphi \mid \exists X \,.\, \varphi \mid \forall x \,.\, \varphi$$

where $a \in A$ and $k \in K$.

- We do not need $\forall X \,.\, \varphi$
- $wMSO(A, K)$: the set of all wMSO-formulas over A and K

Weighted MSO logic - Semantics over finite words

Definition

Let $\varphi \in wMSO(A, K)$. The finitary semantics of φ is the series

$$\|\varphi\| : A^*_{Free(\varphi)} \to K.$$

For every $w \in A^*$ and $(w, Free(\varphi))$-assignment σ, we define $(\|\varphi\|, (w, \sigma))$ inductively by:

- $(\|k\|, (w, \sigma)) = k$

Weighted MSO logic - Semantics over finite words

Definition

Let $\varphi \in wMSO(A, K)$. The finitary semantics of φ is the series

$$\|\varphi\| : A^*_{Free(\varphi)} \to K.$$

For every $w \in A^*$ and $(w, Free(\varphi))$-assignment σ, we define $(\|\varphi\|, (w, \sigma))$ inductively by:

- $(\|k\|, (w, \sigma)) = k$
- $(\|P_a(x)\|, (w, \sigma)) = \begin{cases} 1 & \text{if } w(\sigma(x)) = a \\ 0 & \text{otherwise} \end{cases}$

Weighted MSO logic - Semantics over finite words

Definition

Let $\varphi \in wMSO(A, K)$. The finitary semantics of φ is the series

$$\|\varphi\| : A^*_{Free(\varphi)} \to K.$$

For every $w \in A^*$ and $(w, Free(\varphi))$-assignment σ, we define $(\|\varphi\|, (w, \sigma))$ inductively by:

- $(\|k\|, (w, \sigma)) = k$
- $(\|P_a(x)\|, (w, \sigma)) = \begin{cases} 1 & \text{if } w(\sigma(x)) = a \\ 0 & \text{otherwise} \end{cases}$
- $(\|x \in X\|, (w, \sigma)) = \begin{cases} 1 & \text{if } \sigma(x) \in \sigma(X) \\ 0 & \text{otherwise} \end{cases}$

Weighted MSO logic - Semantics over finite words

Definition

Let $\varphi \in wMSO(A, K)$. The finitary semantics of φ is the series

$$\|\varphi\| : A^*_{Free(\varphi)} \to K.$$

For every $w \in A^*$ and $(w, Free(\varphi))$-assignment σ, we define $(\|\varphi\|, (w, \sigma))$ inductively by:

- $(\|k\|, (w, \sigma)) = k$
- $(\|P_a(x)\|, (w, \sigma)) = \begin{cases} 1 & \text{if } w(\sigma(x)) = a \\ 0 & \text{otherwise} \end{cases}$
- $(\|x \in X\|, (w, \sigma)) = \begin{cases} 1 & \text{if } \sigma(x) \in \sigma(X) \\ 0 & \text{otherwise} \end{cases}$
- $(\|x \leq y\|, (w, \sigma)) = \begin{cases} 1 & \text{if } \sigma(x) \leq \sigma(y) \\ 0 & \text{otherwise} \end{cases}$

Weighted MSO logic - Semantics over finite words

Definition (continued)

- $(\|\neg\varphi\|, (w,\sigma)) = \begin{cases} 1 & \text{if} \quad (\|\varphi\|, (w,\sigma)) = 0 \\ 0 & \text{if} \quad (\|\varphi\|, (w,\sigma)) = 1 \end{cases}$, provided that φ is of the form $P_a(x)$, $x \leq y$ or $x \in X$

Weighted MSO logic - Semantics over finite words

Definition (continued)

- $(\|\neg\varphi\|, (w,\sigma)) = \begin{cases} 1 & \text{if} \quad (\|\varphi\|, (w,\sigma)) = 0 \\ 0 & \text{if} \quad (\|\varphi\|, (w,\sigma)) = 1 \end{cases}$, provided that φ is of the form $P_a(x)$, $x \leq y$ or $x \in X$
- $(\|\varphi \vee \psi\|, (w,\sigma)) = (\|\varphi\|, (w,\sigma)) + (\|\psi\|, (w,\sigma))$

Weighted MSO logic - Semantics over finite words

Definition (continued)

- $(\|\neg\varphi\|, (w,\sigma)) = \begin{cases} 1 & \text{if } (\|\varphi\|, (w,\sigma)) = 0 \\ 0 & \text{if } (\|\varphi\|, (w,\sigma)) = 1 \end{cases}$, provided that φ is of the form $P_a(x)$, $x \leq y$ or $x \in X$
- $(\|\varphi \vee \psi\|, (w,\sigma)) = (\|\varphi\|, (w,\sigma)) + (\|\psi\|, (w,\sigma))$
- $(\|\varphi \wedge \psi\|, (w,\sigma)) = (\|\varphi\|, (w,\sigma)) \cdot (\|\psi\|, (w,\sigma))$

Weighted MSO logic - Semantics over finite words

Definition (continued)

- $(\|\neg\varphi\|, (w,\sigma)) = \begin{cases} 1 & \text{if} \quad (\|\varphi\|, (w,\sigma)) = 0 \\ 0 & \text{if} \quad (\|\varphi\|, (w,\sigma)) = 1 \end{cases}$, provided that φ is of the form $P_a(x)$, $x \leq y$ or $x \in X$
- $(\|\varphi \vee \psi\|, (w,\sigma)) = (\|\varphi\|, (w,\sigma)) + (\|\psi\|, (w,\sigma))$
- $(\|\varphi \wedge \psi\|, (w,\sigma)) = (\|\varphi\|, (w,\sigma)) \cdot (\|\psi\|, (w,\sigma))$
- $(\|\exists x \centerdot \varphi\|, (w,\sigma)) = \sum_{i \in dom(w)} (\|\varphi\|, (w, \sigma[x \to i]))$

Definition (continued)

- $(\|\neg\varphi\|, (w,\sigma)) = \begin{cases} 1 & \text{if} \quad (\|\varphi\|, (w,\sigma)) = 0 \\ 0 & \text{if} \quad (\|\varphi\|, (w,\sigma)) = 1 \end{cases}$, provided that φ is of the form $P_a(x)$, $x \leq y$ or $x \in X$
- $(\|\varphi \vee \psi\|, (w,\sigma)) = (\|\varphi\|, (w,\sigma)) + (\|\psi\|, (w,\sigma))$
- $(\|\varphi \wedge \psi\|, (w,\sigma)) = (\|\varphi\|, (w,\sigma)) \cdot (\|\psi\|, (w,\sigma))$
- $(\|\exists x \,.\, \varphi\|, (w,\sigma)) = \sum_{i \in dom(w)} (\|\varphi\|, (w, \sigma[x \to i]))$
- $(\|\exists X \,.\, \varphi\|, (w,\sigma)) = \sum_{I \subseteq dom(w)} (\|\varphi\|, (w, \sigma[X \to I]))$

Weighted MSO logic - Semantics over finite words

Definition (continued)

- $(\|\neg\varphi\|, (w,\sigma)) = \begin{cases} 1 & \text{if} \quad (\|\varphi\|, (w,\sigma)) = 0 \\ 0 & \text{if} \quad (\|\varphi\|, (w,\sigma)) = 1 \end{cases}$, provided that φ is of the form $P_a(x)$, $x \leq y$ or $x \in X$
- $(\|\varphi \vee \psi\|, (w,\sigma)) = (\|\varphi\|, (w,\sigma)) + (\|\psi\|, (w,\sigma))$
- $(\|\varphi \wedge \psi\|, (w,\sigma)) = (\|\varphi\|, (w,\sigma)) \cdot (\|\psi\|, (w,\sigma))$
- $(\|\exists x . \varphi\|, (w,\sigma)) = \sum_{i \in dom(w)} (\|\varphi\|, (w,\sigma[x \to i]))$
- $(\|\exists X . \varphi\|, (w,\sigma)) = \sum_{I \subseteq dom(w)} (\|\varphi\|, (w,\sigma[X \to I]))$
- $(\|\forall x . \varphi\|, (w,\sigma)) = \prod_{i \in dom(w)} (\|\varphi\|, (w,\sigma[x \to i]))$

Weighted MSO logic - Semantics over finite words

Definition (continued)

- $(\|\neg\varphi\|, (w,\sigma)) = \begin{cases} 1 & \text{if} \quad (\|\varphi\|, (w,\sigma)) = 0 \\ 0 & \text{if} \quad (\|\varphi\|, (w,\sigma)) = 1 \end{cases}$, provided that φ is of the form $P_a(x)$, $x \leq y$ or $x \in X$
- $(\|\varphi \vee \psi\|, (w,\sigma)) = (\|\varphi\|, (w,\sigma)) + (\|\psi\|, (w,\sigma))$
- $(\|\varphi \wedge \psi\|, (w,\sigma)) = (\|\varphi\|, (w,\sigma)) \cdot (\|\psi\|, (w,\sigma))$
- $(\|\exists x \,.\, \varphi\|, (w,\sigma)) = \sum_{i \in dom(w)} (\|\varphi\|, (w, \sigma[x \rightarrow i]))$
- $(\|\exists X \,.\, \varphi\|, (w,\sigma)) = \sum_{I \subseteq dom(w)} (\|\varphi\|, (w, \sigma[X \rightarrow I]))$
- $(\|\forall x \,.\, \varphi\|, (w,\sigma)) = \prod_{i \in dom(w)} (\|\varphi\|, (w, \sigma[x \rightarrow i]))$

- where $dom(w) = \{0, \ldots, |w| - 1\}$

Weighted MSO logic - Semantics over finite words

- If $Free(\varphi) = \emptyset$, then φ is a sentence and $\|\varphi\| : A^* \to K$

- If $Free(\varphi) = \emptyset$, then φ is a sentence and $\|\varphi\| : A^* \to K$
- Example:

Weighted MSO logic - Semantics over finite words

- If $Free(\varphi) = \emptyset$, then φ is a sentence and $\|\varphi\| : A^* \to K$
- Example:
- Let $A = \{a, b, c\}$ and
  $\varphi = \forall x . (((P_a(x) \wedge 1) \vee 0) \wedge ((P_b(x) \wedge 1) \vee 0))$

- If $Free(\varphi) = \emptyset$, then φ is a sentence and $\|\varphi\| : A^* \to K$
- Example:
- Let $A = \{a, b, c\}$ and
  $\varphi = \forall x \, . \, (((P_a(x) \wedge 1) \vee 0) \wedge ((P_b(x) \wedge 1) \vee 0))$
- Consider the semiring $(\mathbb{N}, +, \cdot, 0, 1)$ of natural numbers. Then for every $w \in A^*$

Weighted MSO logic - Semantics over finite words

- If $Free(\varphi) = \emptyset$, then φ is a sentence and $\|\varphi\| : A^* \to K$
- Example:
- Let $A = \{a, b, c\}$ and
  $\varphi = \forall x \,.\, (((P_a(x) \wedge 1) \vee 0) \wedge ((P_b(x) \wedge 1) \vee 0))$
- Consider the semiring $(\mathbb{N}, +, \cdot, 0, 1)$ of natural numbers. Then for every $w \in A^*$
  - $(\|\varphi\|, w) = 0$

Weighted MSO logic - Semantics over finite words

- If $Free(\varphi) = \emptyset$, then φ is a sentence and $\|\varphi\| : A^* \to K$
- Example:
- Let $A = \{a, b, c\}$ and
  $\varphi = \forall x . (((P_a(x) \wedge 1) \vee 0) \wedge ((P_b(x) \wedge 1) \vee 0))$
- Consider the semiring $(\mathbb{N}, +, \cdot, 0, 1)$ of natural numbers. Then for every $w \in A^*$
  - $(\|\varphi\|, w) = 0$
- Now consider the max-plus semiring $(\mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$. For every $w \in A^*$

Weighted MSO logic - Semantics over finite words

- If $Free(\varphi) = \emptyset$, then φ is a sentence and $\|\varphi\| : A^* \to K$
- Example:
- Let $A = \{a, b, c\}$ and
  $\varphi = \forall x.(((P_a(x) \wedge 1) \vee 0) \wedge ((P_b(x) \wedge 1) \vee 0))$
- Consider the semiring $(\mathbb{N}, +, \cdot, 0, 1)$ of natural numbers. Then for every $w \in A^*$
  - $(\|\varphi\|, w) = 0$
- Now consider the max-plus semiring $(\mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$. For every $w \in A^*$
  - $(\|\varphi\|, w) = |w|_a + |w|_b$

Recognizability and definability

- A series $s : A^* \longrightarrow K$ is called *wMSO-definable* if there is a wMSO-sentence φ over A and K so that $s = \|\varphi\|$

Recognizability and definability

- A series $s : A^* \longrightarrow K$ is called *wMSO-definable* if there is a wMSO-sentence φ over A and K so that $s = \|\varphi\|$
- $wMso(A, K)$: the class of all wMSO-definable series over A and K

- A series $s : A^* \longrightarrow K$ is called *wMSO-definable* if there is a wMSO-sentence φ over A and K so that $s = \|\varphi\|$
- $wMso(A, K)$: the class of all wMSO-definable series over A and K

Theorem (Droste & Gastin 2005)

Recognizability and definability

- A series $s : A^* \longrightarrow K$ is called *wMSO-definable* if there is a wMSO-sentence φ over A and K so that $s = \|\varphi\|$
- $wMso(A, K)$: the class of all wMSO-definable series over A and K

Theorem (Droste & Gastin 2005)

- $Rec(A, K) \subsetneq wMso(A, K)$

Recognizability and definability

- A series $s : A^* \to K$ is called *wMSO-definable* if there is a wMSO-sentence φ over A and K so that $s = \|\varphi\|$
- $wMso(A, K)$: the class of all wMSO-definable series over A and K

Theorem (Droste & Gastin 2005)

- $Rec(A, K) \subsetneq wMso(A, K)$
- $Rec(A, K) =$ *a fragment of* $wMso(A, K)$ *(Büchi-type theorem)*

Recognizability and definability

- A series $s : A^* \longrightarrow K$ is called *wMSO-definable* if there is a wMSO-sentence φ over A and K so that $s = \|\varphi\|$
- $wMso(A, K)$: the class of all wMSO-definable series over A and K

Theorem (Droste & Gastin 2005)

- $Rec(A, K) \subsetneq wMso(A, K)$
- $Rec(A, K)$ = *a fragment of* $wMso(A, K)$ *(Büchi-type theorem)*
- *If K is locally finite, i.e., the subsemiring generated by any finite subset of K is finite, then $Rec(A, K) = wMso(A, K)$*

Recognizability and definability

- A series $s : A^* \longrightarrow K$ is called *wMSO-definable* if there is a wMSO-sentence φ over A and K so that $s = \|\varphi\|$
- $wMso(A, K)$: the class of all wMSO-definable series over A and K

Theorem (Droste & Gastin 2005)

- $Rec(A, K) \subsetneq wMso(A, K)$
- $Rec(A, K)$ = *a fragment of* $wMso(A, K)$ *(Büchi-type theorem)*
- *If K is locally finite, i.e., the subsemiring generated by any finite subset of K is finite, then* $Rec(A, K) = wMso(A, K)$

- Open: $wMso(A, K) =?$

Weighted MSO logic - Semantics over infinite words

Definition

Let $\varphi \in wMSO(A, K)$. The infinitary semantics of φ is the series

$$\|\varphi\| : A^{\omega}_{Free(\varphi)} \to K.$$

For every $w \in A^*$ and $(w, Free(\varphi))$-assignment σ, we define $(\|\varphi\|, (w, \sigma))$ inductively by:

- $(\|k\|, (w, \sigma)) = k$
- $(\|P_a(x)\|, (w, \sigma)) = \begin{cases} 1 & \text{if } w(\sigma(x)) = a \\ 0 & \text{otherwise} \end{cases}$
- $(\|x \in X\|, (w, \sigma)) = \begin{cases} 1 & \text{if } \sigma(x) \in \sigma(X) \\ 0 & \text{otherwise} \end{cases}$
- $(\|x \leq y\|, (w, \sigma)) = \begin{cases} 1 & \text{if } \sigma(x) \leq \sigma(y) \\ 0 & \text{otherwise} \end{cases}$

Weighted MSO logic - Semantics over infinite words

Definition

- $(\|\neg\varphi\|, (w,\sigma)) = \begin{cases} 1 & \text{if} \quad (\|\varphi\|, (w,\sigma)) = 0 \\ 0 & \text{if} \quad (\|\varphi\|, (w,\sigma)) = 1 \end{cases}$, provided that φ is of the form $P_a(x)$, $x \leq y$ or $x \in X$
- $(\|\varphi \vee \psi\|, (w,\sigma)) = (\|\varphi\|, (w,\sigma)) + (\|\psi\|, (w,\sigma))$
- $(\|\varphi \wedge \psi\|, (w,\sigma)) = (\|\varphi\|, (w,\sigma)) \cdot (\|\psi\|, (w,\sigma))$
- $(\|\exists x \,.\, \varphi\|, (w,\sigma)) = \sum_{i \in dom(w)} (\|\varphi\|, (w,\sigma[x \to i]))$
- $(\|\exists X \,.\, \varphi\|, (w,\sigma)) = \sum_{I \subseteq dom(w)} (\|\varphi\|, (w,\sigma[X \to I]))$
- $(\|\forall x \,.\, \varphi\|, (w,\sigma)) = \prod_{i \in dom(w)} (\|\varphi\|, (w,\sigma[x \to i]))$

- where $dom(w) = \omega$

Recognizability and definability

- If $Free(\varphi) = \emptyset$, then φ is a sentence and $\|\varphi\| : A^\omega \to K$
- An infinitary series $s : A^\omega \to K$ is called *wMSO-definable* if there is a wMSO-sentence φ over A and K so that $s = \|\varphi\|$
- ω-$wMso(A, K)$: the class of all infinitary wMSO-definable series over A and K

- Büchi type theorem:

Recognizability and definability

- If $Free(\varphi) = \emptyset$, then φ is a sentence and $\|\varphi\| : A^\omega \to K$
- An infinitary series $s : A^\omega \to K$ is called *wMSO-definable* if there is a wMSO-sentence φ over A and K so that $s = \|\varphi\|$
- ω-$wMso(A, K)$: the class of all infinitary wMSO-definable series over A and K

- Büchi type theorem:

Theorem (Droste & R 2006)

$$\omega\text{-}Rec(A, K) = \textit{a fragment of } \omega\text{-}wMso(A, K)$$

Recognizability and definability

- If $Free(\varphi) = \emptyset$, then φ is a sentence and $\|\varphi\| : A^\omega \to K$
- An infinitary series $s : A^\omega \to K$ is called *wMSO-definable* if there is a wMSO-sentence φ over A and K so that $s = \|\varphi\|$
- $\omega\text{-}wMso(A, K)$: the class of all infinitary wMSO-definable series over A and K

- Büchi type theorem:

Theorem (Droste & R 2006)

$$\omega\text{-}Rec(A, K) = a\ fragment\ of\ \omega\text{-}wMso(A, K)$$

- Open:

Recognizability and definability

- If $Free(\varphi) = \emptyset$, then φ is a sentence and $\|\varphi\| : A^\omega \to K$
- An infinitary series $s : A^\omega \to K$ is called *wMSO-definable* if there is a wMSO-sentence φ over A and K so that $s = \|\varphi\|$
- ω-$wMso(A, K)$: the class of all infinitary wMSO-definable series over A and K

- Büchi type theorem:

Theorem (Droste & R 2006)

$$\omega\text{-}Rec(A, K) = \text{a fragment of } \omega\text{-}wMso(A, K)$$

- Open:
  - ω-$Rec(A, K) \subseteq \omega$-$wMso(A, K)$ is the inclusion proper? (guess: Yes)

Recognizability and definability

- If $Free(\varphi) = \emptyset$, then φ is a sentence and $\|\varphi\| : A^\omega \to K$
- An infinitary series $s : A^\omega \to K$ is called *wMSO-definable* if there is a wMSO-sentence φ over A and K so that $s = \|\varphi\|$
- ω-$wMso(A, K)$: the class of all infinitary wMSO-definable series over A and K

- Büchi type theorem:

Theorem (Droste & R 2006)

$$\omega\text{-}Rec(A, K) = a \text{ fragment of } \omega\text{-}wMso(A, K)$$

- Open:
  - ω-$Rec(A, K) \subseteq \omega$-$wMso(A, K)$ is the inclusion proper? (guess: Yes)
  - ω-$wMso(A, K) = ?$

Automata and logic over the max-plus and min-plus semirings

- $\mathbb{R}_{\max} = (\mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$ the *max-plus semiring*

Automata and logic over the max-plus and min-plus semirings

- $\mathbb{R}_{\max} = (\mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$ the *max-plus semiring*
- $\mathbb{R}_{\min} = (\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ the *min-plus semiring*

Automata and logic over the max-plus and min-plus semirings

- $\mathbb{R}_{\max} = (\mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$ the *max-plus semiring*
- $\mathbb{R}_{\min} = (\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ the *min-plus semiring*
- Why should we consider weighted automata and wMSO logic over $\mathbb{R}_{\max}$ and $\mathbb{R}_{\min}$?

Automata and logic over the max-plus and min-plus semirings

- $\mathbb{R}_{\max} = (\mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$ the *max-plus semiring*
- $\mathbb{R}_{\min} = (\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ the *min-plus semiring*
- Why should we consider weighted automata and wMSO logic over $\mathbb{R}_{\max}$ and $\mathbb{R}_{\min}$?
- Zimmermann 1981: applications in optimization problems

Automata and logic over the max-plus and min-plus semirings

- $\mathbb{R}_{\max} = (\mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$ the *max-plus semiring*
- $\mathbb{R}_{\min} = (\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ the *min-plus semiring*
- Why should we consider weighted automata and wMSO logic over $\mathbb{R}_{\max}$ and $\mathbb{R}_{\min}$?
- Zimmermann 1981: applications in optimization problems
- Consider a weighted Büchi automaton $\mathcal{A} = (Q, A, in, wt, F)$, a word $w = a_0 a_1 \ldots \in A^\omega$ and a path $P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots$ of $\mathcal{A}$ over w. Then we should have

$$weight(P_w) = in(q_0) + \sum_{i \geq 0} wt((q_i, a_i, q_{i+1}))$$

Automata and logic over the max-plus and min-plus semirings

- $\mathbb{R}_{\max} = (\mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$ the *max-plus semiring*
- $\mathbb{R}_{\min} = (\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ the *min-plus semiring*
- Why should we consider weighted automata and wMSO logic over $\mathbb{R}_{\max}$ and $\mathbb{R}_{\min}$?
- Zimmermann 1981: applications in optimization problems
- Consider a weighted Büchi automaton $\mathcal{A} = (Q, A, in, wt, F)$, a word $w = a_0 a_1 \ldots \in A^\omega$ and a path $P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots$ of $\mathcal{A}$ over w. Then we should have

$$weight(P_w) = in(q_0) + \sum_{i \geq 0} wt((q_i, a_i, q_{i+1}))$$

- but this infinite sum does not always exist!

Automata and logic over the max-plus and min-plus semirings

- $\mathbb{R}_{\max} = (\mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$ the *max-plus semiring*
- $\mathbb{R}_{\min} = (\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ the *min-plus semiring*
- Why should we consider weighted automata and wMSO logic over $\mathbb{R}_{\max}$ and $\mathbb{R}_{\min}$?
- Zimmermann 1981: applications in optimization problems
- Consider a weighted Büchi automaton $\mathcal{A} = (Q, A, in, wt, F)$, a word $w = a_0 a_1 \ldots \in A^\omega$ and a path $P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots$ of $\mathcal{A}$ over w. Then we should have

$$weight(P_w) = in(q_0) + \sum_{i \geq 0} wt((q_i, a_i, q_{i+1}))$$

- but this infinite sum does not always exist!
- Solution: *discounting*

Automata and logic over the max-plus and min-plus semirings

- $\mathbb{R}_{\max} = (\mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$ the *max-plus semiring*
- $\mathbb{R}_{\min} = (\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ the *min-plus semiring*
- Why should we consider weighted automata and wMSO logic over $\mathbb{R}_{\max}$ and $\mathbb{R}_{\min}$?
- Zimmermann 1981: applications in optimization problems
- Consider a weighted Büchi automaton $\mathcal{A} = (Q, A, in, wt, F)$, a word $w = a_0 a_1 \ldots \in A^\omega$ and a path $P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots$ of $\mathcal{A}$ over w. Then we should have

$$weight(P_w) = in(q_0) + \sum_{i \geq 0} wt((q_i, a_i, q_{i+1}))$$

- but this infinite sum does not always exist!
- Solution: *discounting*
- Motivation

Automata and logic over the max-plus and min-plus semirings

- $\mathbb{R}_{\max} = (\mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$ the *max-plus semiring*
- $\mathbb{R}_{\min} = (\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ the *min-plus semiring*
- Why should we consider weighted automata and wMSO logic over $\mathbb{R}_{\max}$ and $\mathbb{R}_{\min}$?
- Zimmermann 1981: applications in optimization problems
- Consider a weighted Büchi automaton $\mathcal{A} = (Q, A, in, wt, F)$, a word $w = a_0 a_1 \ldots \in A^\omega$ and a path $P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots$ of $\mathcal{A}$ over w. Then we should have

$$weight(P_w) = in(q_0) + \sum_{i \geq 0} wt((q_i, a_i, q_{i+1}))$$

- but this infinite sum does not always exist!
- Solution: *discounting*
- Motivation
    - used in model checking (Henzinger et al 2003, Faella et al 2008)

Automata and logic over the max-plus and min-plus semirings

- $\mathbb{R}_{\max} = (\mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$ the *max-plus semiring*
- $\mathbb{R}_{\min} = (\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ the *min-plus semiring*
- Why should we consider weighted automata and wMSO logic over $\mathbb{R}_{\max}$ and $\mathbb{R}_{\min}$?
- Zimmermann 1981: applications in optimization problems
- Consider a weighted Büchi automaton $\mathcal{A} = (Q, A, in, wt, F)$, a word $w = a_0 a_1 \ldots \in A^\omega$ and a path $P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots$ of $\mathcal{A}$ over w. Then we should have

$$weight(P_w) = in(q_0) + \sum_{i \geq 0} wt((q_i, a_i, q_{i+1}))$$

- but this infinite sum does not always exist!
- Solution: *discounting*
- Motivation
  - used in model checking (Henzinger et al 2003, Faella et al 2008)
  - common in economical mathematics

Weighted Büchi automata with discounting

- $0 \leq d < 1$ a *discounting parameter*

Weighted Büchi automata with discounting

- $0 \leq d < 1$ a *discounting parameter*
- A weighted Büchi automaton $\mathcal{A} = (Q, A, in, wt, F)$, a word $w = a_0 a_1 \ldots \in A^\omega$ and a path $P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots$ of $\mathcal{A}$ over w

Weighted Büchi automata with discounting

- $0 \leq d < 1$ a *discounting parameter*
- A weighted Büchi automaton $\mathcal{A} = (Q, A, in, wt, F)$, a word $w = a_0 a_1 \ldots \in A^\omega$ and a path $P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots$ of $\mathcal{A}$ over w
- The *d-weight* of P_w

$$d\text{-}weight(P_w) = in(q_0) + \sum_{i \geq 0} d^i \cdot wt((q_i, a_i, q_{i+1}))$$

Weighted Büchi automata with discounting

- $0 \leq d < 1$ a *discounting parameter*
- A weighted Büchi automaton $\mathcal{A} = (Q, A, in, wt, F)$, a word $w = a_0 a_1 \ldots \in A^\omega$ and a path $P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots$ of $\mathcal{A}$ over w
- The *d-weight* of P_w

$$d\text{-}weight(P_w) = in(q_0) + \sum_{i \geq 0} d^i \cdot wt((q_i, a_i, q_{i+1}))$$

- This sum exists: let $C = \max\{in(q), wt(t) \mid q \in Q, t \in Q \times A \times Q\}$

Weighted Büchi automata with discounting

- $0 \leq d < 1$ a *discounting parameter*
- A weighted Büchi automaton $\mathcal{A} = (Q, A, in, wt, F)$, a word $w = a_0 a_1 \ldots \in A^\omega$ and a path $P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \ldots$ of $\mathcal{A}$ over w
- The *d-weight* of P_w

$$d\text{-}weight(P_w) = in(q_0) + \sum_{i \geq 0} d^i \cdot wt((q_i, a_i, q_{i+1}))$$

- This sum exists: let $C = \max\{in(q), wt(t) \mid q \in Q, t \in Q \times A \times Q\}$
- $d\text{-}weight(P_w) \leq C + C \cdot \frac{1}{1-d} < \infty$

Weighted Büchi automata with discounting

- *d-behavior of* $\mathcal{A}$:

$$\|\mathcal{A}\|_d : A^\omega \to \mathbb{R}_{\max},$$

where for every $w \in A^\omega$

$$(\|\mathcal{A}\|_d, w) = \sup_{P_w \text{ successful}} (d\text{-}weight(P_w))$$

Weighted Büchi automata with discounting

- *d-behavior of* $\mathcal{A}$:

$$\|\mathcal{A}\|_d : A^\omega \longrightarrow \mathbb{R}_{\max},$$

where for every $w \in A^\omega$

$$(\|\mathcal{A}\|_d, w) = \sup_{P_w \text{ successful}} (d\text{-}weight(P_w))$$

- A series $s : A^\omega \longrightarrow \mathbb{R}_{\max}$ is called *d-ω-recognizable* if there exists a weighted Büchi automaton over A and $\mathbb{R}_{\max}$, so that $s = \|\mathcal{A}\|_d$

Weighted Büchi automata with discounting

- *d-behavior of* $\mathcal{A}$:

$$\|\mathcal{A}\|_d : A^\omega \to \mathbb{R}_{\max},$$

where for every $w \in A^\omega$

$$(\|\mathcal{A}\|_d, w) = \sup_{P_w \text{ successful}} (d\text{-}weight(P_w))$$

- A series $s : A^\omega \to \mathbb{R}_{\max}$ is called *d-ω-recognizable* if there exists a weighted Büchi automaton over A and $\mathbb{R}_{\max}$, so that $s = \|\mathcal{A}\|_d$
- $\omega\text{-}Rec(A, \mathbb{R}_{\max}, d)$: the class of all d-ω-recognizable series over A and $\mathbb{R}_{\max}$

wMSO logic with discounting - d-semantics

Same syntax like in other wMSO

Definition

Let $\varphi \in wMSO(A, \mathbb{R}_{\max})$. The infinitary d-semantics of φ is the series

$$\|\varphi\|_d : A^{\omega}_{Free(\varphi)} \longrightarrow \mathbb{R}_{\max}.$$

For every $w \in A^*$ and $(w, Free(\varphi))$-assignment σ, we define $(\|\varphi\|_d, (w, \sigma))$ inductively by:

- $(\|k\|_d, (w, \sigma)) = k$
- $(\|P_a(x)\|_d, (w, \sigma)) = \begin{cases} 0 & \text{if } w(\sigma(x)) = a \\ -\infty & \text{otherwise} \end{cases}$
- $(\|x \in X\|_d, (w, \sigma)) = \begin{cases} 0 & \text{if } \sigma(x) \in \sigma(X) \\ -\infty & \text{otherwise} \end{cases}$
- $(\|x \leq y\|_d, (w, \sigma)) = \begin{cases} 0 & \text{if } \sigma(x) \leq \sigma(y) \\ -\infty & \text{otherwise} \end{cases}$

wMSO logic with discounting - d-semantics

Definition

- $(\|\neg\varphi\|_d, (w,\sigma)) = \begin{cases} 0 & \text{if } (\|\varphi\|_d, (w,\sigma)) = -\infty \\ -\infty & \text{if } (\|\varphi\|_d, (w,\sigma)) = 0 \end{cases}$, provided that φ is of the form $P_a(x)$, $x \leq y$ or $x \in X$
- $(\|\varphi \vee \psi\|_d, (w,\sigma)) = \max((\|\varphi\|_d, (w,\sigma)), (\|\psi\|_d, (w,\sigma)))$
- $(\|\varphi \wedge \psi\|_d, (w,\sigma)) = (\|\varphi\|_d, (w,\sigma)) + (\|\psi\|_d, (w,\sigma))$
- $(\|\exists x . \varphi\|_d, (w,\sigma)) = \sup_{i \in dom(w)} ((\|\varphi\|_d, (w, \sigma[x \to i])))$
- $(\|\exists X . \varphi\|_d, (w,\sigma)) = \sup_{I \subseteq dom(w)} ((\|\varphi\|_d, (w, \sigma[X \to I])))$
- $(\|\forall x . \varphi\|_d, (w,\sigma)) = \sum_{i \in dom(w)} d^i \cdot (\|\varphi\|_d, (w, \sigma[x \to i]))$

- where $dom(w) = \omega$

wMSO logic with discounting - d-semantics

Definition

- $(\|\neg\varphi\|_d, (w,\sigma)) = \begin{cases} 0 & \text{if } (\|\varphi\|_d, (w,\sigma)) = -\infty \\ -\infty & \text{if } (\|\varphi\|_d, (w,\sigma)) = 0 \end{cases}$, provided that φ is of the form $P_a(x)$, $x \leq y$ or $x \in X$
- $(\|\varphi \vee \psi\|_d, (w,\sigma)) = \max((\|\varphi\|_d, (w,\sigma)), (\|\psi\|_d, (w,\sigma)))$
- $(\|\varphi \wedge \psi\|_d, (w,\sigma)) = (\|\varphi\|_d, (w,\sigma)) + (\|\psi\|_d, (w,\sigma))$
- $(\|\exists x \,.\, \varphi\|_d, (w,\sigma)) = \sup_{i \in dom(w)} ((\|\varphi\|_d, (w, \sigma[x \to i])))$
- $(\|\exists X \,.\, \varphi\|_d, (w,\sigma)) = \sup_{I \subseteq dom(w)} ((\|\varphi\|_d, (w, \sigma[X \to I])))$
- $(\|\forall x \,.\, \varphi\|_d, (w,\sigma)) = \sum_{i \in dom(w)} d^i \cdot (\|\varphi\|_d, (w, \sigma[x \to i]))$

- where $dom(w) = \omega$

d-recognizability and d-definability

- An infinitary series $s : A^\omega \to \mathbb{R}_{\max}$ is called *wMSO-d-definable* if there is a wMSO-sentence φ over A and $\mathbb{R}_{\max}$ so that $s = \|\varphi\|_d$

d-recognizability and d-definability

- An infinitary series $s : A^\omega \to \mathbb{R}_{\max}$ is called *wMSO-d-definable* if there is a wMSO-sentence φ over A and $\mathbb{R}_{\max}$ so that $s = \|\varphi\|_d$
- ω-*wMso*$(A, \mathbb{R}_{\max}, d)$: the class of all infinitary wMSO-d-definable series over A and $\mathbb{R}_{\max}$

d-recognizability and d-definability

- An infinitary series $s : A^\omega \to \mathbb{R}_{\max}$ is called *wMSO-d-definable* if there is a wMSO-sentence φ over A and $\mathbb{R}_{\max}$ so that $s = \|\varphi\|_d$
- ω-*wMso*$(A, \mathbb{R}_{\max}, d)$: the class of all infinitary wMSO-d-definable series over A and $\mathbb{R}_{\max}$
- Büchi type theorem:

- An infinitary series $s : A^\omega \to \mathbb{R}_{\max}$ is called *wMSO-d-definable* if there is a wMSO-sentence φ over A and $\mathbb{R}_{\max}$ so that $s = \|\varphi\|_d$
- ω-$wMso(A, \mathbb{R}_{\max}, d)$: the class of all infinitary wMSO-d-definable series over A and $\mathbb{R}_{\max}$
- Büchi type theorem:

Theorem (Droste & R 2007)

$$\omega\text{-}Rec(A, \mathbb{R}_{\max}, d) = \textit{a fragment of } \omega\text{-}wMso(A, \mathbb{R}_{\max}, d)$$

d-recognizability and d-definability

- An infinitary series $s : A^\omega \to \mathbb{R}_{\max}$ is called *wMSO-d-definable* if there is a wMSO-sentence φ over A and $\mathbb{R}_{\max}$ so that $s = \|\varphi\|_d$
- ω-$wMso(A, \mathbb{R}_{\max}, d)$: the class of all infinitary wMSO-d-definable series over A and $\mathbb{R}_{\max}$
- Büchi type theorem:

Theorem (Droste & R 2007)

$$\omega\text{-}Rec(A, \mathbb{R}_{\max}, d) = \textit{a fragment of } \omega\text{-}wMso(A, \mathbb{R}_{\max}, d)$$

- Open:

d-recognizability and d-definability

- An infinitary series $s : A^\omega \to \mathbb{R}_{max}$ is called *wMSO-d-definable* if there is a wMSO-sentence φ over A and $\mathbb{R}_{max}$ so that $s = \|\varphi\|_d$
- $\omega\text{-}wMso(A, \mathbb{R}_{max}, d)$: the class of all infinitary wMSO-d-definable series over A and $\mathbb{R}_{max}$
- Büchi type theorem:

Theorem (Droste & R 2007)

$$\omega\text{-}Rec(A, \mathbb{R}_{max}, d) = \textit{a fragment of } \omega\text{-}wMso(A, \mathbb{R}_{max}, d)$$

- Open:
  - $\omega\text{-}Rec(A, \mathbb{R}_{max}, d) \subseteq \omega\text{-}wMso(A, \mathbb{R}_{max}, d)$ is the inclusion proper? (guess: Yes)

d-recognizability and d-definability

- An infinitary series $s : A^\omega \rightarrow \mathbb{R}_{\max}$ is called *wMSO-d-definable* if there is a wMSO-sentence φ over A and $\mathbb{R}_{\max}$ so that $s = \|\varphi\|_d$
- $\omega\text{-}wMso(A, \mathbb{R}_{\max}, d)$: the class of all infinitary wMSO-d-definable series over A and $\mathbb{R}_{\max}$
- Büchi type theorem:

Theorem (Droste & R 2007)

$$\omega\text{-}Rec(A, \mathbb{R}_{\max}, d) = \textit{a fragment of } \omega\text{-}wMso(A, \mathbb{R}_{\max}, d)$$

- Open:
  - $\omega\text{-}Rec(A, \mathbb{R}_{\max}, d) \subseteq \omega\text{-}wMso(A, \mathbb{R}_{\max}, d)$ is the inclusion proper? (guess: Yes)
  - $\omega\text{-}wMso(A, \mathbb{R}_{\max}, d) = ?$

Linear Temporal Logic (LTL) - Motivation

- Why we are *still* interested in LTL?

Linear Temporal Logic (LTL) - Motivation

- Why we are *still* interested in LTL?
- The IEEE standarized *Propert Spesification Language* (*PSL*) is an extension of LTL, and is increasingly used in many steps of the hardware design, from specification to verification

Linear Temporal Logic (LTL) - Motivation

- Why we are *still* interested in LTL?
- The IEEE standarized *Propert Spesification Language* (*PSL*) is an extension of LTL, and is increasingly used in many steps of the hardware design, from specification to verification
- Version of PSL used in the industry

Linear Temporal Logic (LTL) - Motivation

- Why we are *still* interested in LTL?
- The IEEE standarized *Propert Spesification Language* (*PSL*) is an extension of LTL, and is increasingly used in many steps of the hardware design, from specification to verification
- Version of PSL used in the industry
    - *CBV* from Motorola

Linear Temporal Logic (LTL) - Motivation

- Why we are *still* interested in LTL?
- The IEEE standarized *Propert Spesification Language* (*PSL*) is an extension of LTL, and is increasingly used in many steps of the hardware design, from specification to verification
- Version of PSL used in the industry
  - *CBV* from Motorola
  - *ForSpec* from Intel

Linear Temporal Logic (LTL) - Motivation

- Why we are *still* interested in LTL?
- The IEEE standarized *Propert Spesification Language* (*PSL*) is an extension of LTL, and is increasingly used in many steps of the hardware design, from specification to verification
- Version of PSL used in the industry
  - *CBV* from Motorola
  - *ForSpec* from Intel
  - *Temporal* – *e* from Versity

Linear Temporal Logic (LTL) - Motivation

- Why we are *still* interested in LTL?
- The IEEE standarized *Propert Spesification Language* (*PSL*) is an extension of LTL, and is increasingly used in many steps of the hardware design, from specification to verification
- Version of PSL used in the industry
  - *CBV* from Motorola
  - *ForSpec* from Intel
  - *Temporal* – *e* from Versity
  - *Sugar* from IBM.

Definition

Let AP be a finite set of atomic propositions. The syntax of the LTL-formulas over AP is given by

$$\varphi ::= true \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \varphi U \varphi \mid \Box\varphi \mid \Diamond\varphi \mid \Box\Diamond\varphi$$

where $p \in AP$.

LTL - Syntax

Definition

Let AP be a finite set of atomic propositions. The syntax of the LTL-formulas over AP is given by

$$\varphi ::= true \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \varphi U \varphi \mid \Box\varphi \mid \Diamond\varphi \mid \Box\Diamond\varphi$$

where $p \in AP$.

- $LTL(AP)$: the set of all LTL-formulas over AP.

LTL - Semantics

- Let $\varphi \in LTL(AP)$ and $w = a_0 a_1 a_2 \ldots \in \left(2^{AP}\right)^{\omega}$. We define the satisfaction $w \models \varphi$ of φ by w by induction on the structure of φ:

LTL - Semantics

- Let $\varphi \in LTL(AP)$ and $w = a_0 a_1 a_2 \ldots \in \left(2^{AP}\right)^{\omega}$. We define the satisfaction $w \models \varphi$ of φ by w by induction on the structure of φ:
  - $w \models true$

LTL - Semantics

- Let $\varphi \in LTL(AP)$ and $w = a_0 a_1 a_2 \ldots \in \left(2^{AP}\right)^{\omega}$. We define the satisfaction $w \models \varphi$ of φ by w by induction on the structure of φ:
  - $w \models true$
  - $w \models p$ iff $p \in a_0$

LTL - Semantics

- Let $\varphi \in LTL(AP)$ and $w = a_0 a_1 a_2 \ldots \in \left(2^{AP}\right)^{\omega}$. We define the satisfaction $w \models \varphi$ of φ by w by induction on the structure of φ:
  - $w \models true$
  - $w \models p$ iff $p \in a_0$
  - $w \models \neg\varphi$ iff $w \not\models \varphi$

LTL - Semantics

- Let $\varphi \in LTL(AP)$ and $w = a_0 a_1 a_2 \ldots \in \left(2^{AP}\right)^{\omega}$. We define the satisfaction $w \models \varphi$ of φ by w by induction on the structure of φ:
  - $w \models true$
  - $w \models p$ iff $p \in a_0$
  - $w \models \neg\varphi$ iff $w \not\models \varphi$
  - $w \models \varphi \vee \psi$ iff $w \models \varphi$ or $w \models \psi$

LTL - Semantics

- Let $\varphi \in LTL(AP)$ and $w = a_0 a_1 a_2 \ldots \in \left(2^{AP}\right)^{\omega}$. We define the satisfaction $w \models \varphi$ of φ by w by induction on the structure of φ:
  - $w \models true$
  - $w \models p$ iff $p \in a_0$
  - $w \models \neg\varphi$ iff $w \not\models \varphi$
  - $w \models \varphi \vee \psi$ iff $w \models \varphi$ or $w \models \psi$
  - $w \models \bigcirc\varphi$ iff $a_1 a_2 \ldots \models \varphi$

LTL - Semantics

- Let $\varphi \in LTL(AP)$ and $w = a_0 a_1 a_2 \ldots \in \left(2^{AP}\right)^{\omega}$. We define the satisfaction $w \models \varphi$ of φ by w by induction on the structure of φ:
  - $w \models true$
  - $w \models p$ iff $p \in a_0$
  - $w \models \neg\varphi$ iff $w \not\models \varphi$
  - $w \models \varphi \vee \psi$ iff $w \models \varphi$ or $w \models \psi$
  - $w \models \bigcirc\varphi$ iff $a_1 a_2 \ldots \models \varphi$
  - $w \models \varphi U \psi$ iff $\exists j \geq 0$, $a_j a_{j+1} \ldots \models \psi$ and for every $0 \leq i < j$, $a_i a_{i+1} \ldots \models \varphi$

LTL - Semantics

- Let $\varphi \in LTL(AP)$ and $w = a_0 a_1 a_2 \ldots \in \left(2^{AP}\right)^{\omega}$. We define the satisfaction $w \models \varphi$ of φ by w by induction on the structure of φ:
  - $w \models true$
  - $w \models p$ iff $p \in a_0$
  - $w \models \neg\varphi$ iff $w \not\models \varphi$
  - $w \models \varphi \vee \psi$ iff $w \models \varphi$ or $w \models \psi$
  - $w \models \bigcirc\varphi$ iff $a_1 a_2 \ldots \models \varphi$
  - $w \models \varphi U \psi$ iff $\exists j \geq 0$, $a_j a_{j+1} \ldots \models \psi$ and for every $0 \leq i < j$, $a_i a_{i+1} \ldots \models \varphi$
  - $w \models \Box\varphi$ iff $a_i a_{i+1} \ldots \models \varphi$ for every $i \geq 0$

LTL - Semantics

- Let $\varphi \in LTL(AP)$ and $w = a_0 a_1 a_2 \ldots \in \left(2^{AP}\right)^{\omega}$. We define the satisfaction $w \models \varphi$ of φ by w by induction on the structure of φ:
  - $w \models true$
  - $w \models p$ iff $p \in a_0$
  - $w \models \neg\varphi$ iff $w \not\models \varphi$
  - $w \models \varphi \vee \psi$ iff $w \models \varphi$ or $w \models \psi$
  - $w \models \bigcirc\varphi$ iff $a_1 a_2 \ldots \models \varphi$
  - $w \models \varphi U \psi$ iff $\exists j \geq 0$, $a_j a_{j+1} \ldots \models \psi$ and for every $0 \leq i < j$, $a_i a_{i+1} \ldots \models \varphi$
  - $w \models \Box\varphi$ iff $a_i a_{i+1} \ldots \models \varphi$ for every $i \geq 0$
  - $w \models \Diamond\varphi$ iff $\exists i \geq 0$, $a_i a_{i+1} \ldots \models \varphi$

LTL - Semantics

- Let $\varphi \in LTL(AP)$ and $w = a_0 a_1 a_2 \ldots \in \left(2^{AP}\right)^{\omega}$. We define the satisfaction $w \models \varphi$ of φ by w by induction on the structure of φ:
  - $w \models true$
  - $w \models p$ iff $p \in a_0$
  - $w \models \neg\varphi$ iff $w \not\models \varphi$
  - $w \models \varphi \vee \psi$ iff $w \models \varphi$ or $w \models \psi$
  - $w \models \bigcirc\varphi$ iff $a_1 a_2 \ldots \models \varphi$
  - $w \models \varphi U \psi$ iff $\exists j \geq 0$, $a_j a_{j+1} \ldots \models \psi$ and for every $0 \leq i < j$, $a_i a_{i+1} \ldots \models \varphi$
  - $w \models \Box\varphi$ iff $a_i a_{i+1} \ldots \models \varphi$ for every $i \geq 0$
  - $w \models \Diamond\varphi$ iff $\exists i \geq 0$, $a_i a_{i+1} \ldots \models \varphi$
  - $w \models \Box\Diamond\varphi$ iff for every $i \geq 0$, $\exists j \geq i$ such that $a_j a_{j+1} \ldots \models \varphi$.

LTL-definability and recognizability

- $\varphi \in LTL(AP)$

LTL-definability and recognizability

- $\varphi \in LTL(AP)$
- $L(\varphi)$: the *language of* (all infinite words over 2^{AP} satisfying) φ

LTL-definability and recognizability

- $\varphi \in LTL(AP)$
- $L(\varphi)$: the *language of* (all infinite words over 2^{AP} satisfying) φ
- $L \subseteq (2^{AP})^{\omega}$ is *LTL-definable* if there is a $\varphi \in LTL(AP)$ such that $L = L(\varphi)$

LTL-definability and recognizability

- $\varphi \in LTL(AP)$
- $L(\varphi)$: the *language of* (all infinite words over 2^{AP} satisfying) φ
- $L \subseteq (2^{AP})^{\omega}$ is *LTL-definable* if there is a $\varphi \in LTL(AP)$ such that $L = L(\varphi)$
- ω-$Ltl(2^{AP})$: the class of all LTL-definable infinitary languages over 2^{AP}

LTL-definability and recognizability

- $\varphi \in LTL(AP)$
- $L(\varphi)$: the *language of* (all infinite words over 2^{AP} satisfying) φ
- $L \subseteq (2^{AP})^\omega$ is *LTL-definable* if there is a $\varphi \in LTL(AP)$ such that $L = L(\varphi)$
- $\omega\text{-}Ltl(2^{AP})$: the class of all LTL-definable infinitary languages over 2^{AP}
- Vardi and Wopler 1994:

$$\omega\text{-}Ltl(2^{AP}) \subsetneq \omega\text{-}Rec(2^{AP})$$

wLTL - Syntax

Definition

Let AP be a finite set of atomic propositions. The syntax of the wLTL-formulas with discounting over AP and $\mathbb{R}_{\max}$ is given by

$$\varphi ::= k \mid p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc \varphi \mid \varphi U \varphi \mid \Box \varphi \mid \Diamond \varphi \mid \Box \Diamond \varphi$$

where $k \in \mathbb{R}_{\max}$ and $p \in AP$.

wLTL - Syntax

Definition

Let AP be a finite set of atomic propositions. The syntax of the wLTL-formulas with discounting over AP and $\mathbb{R}_{\max}$ is given by

$$\varphi ::= k \mid p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi U \varphi \mid \Box\varphi \mid \Diamond\varphi \mid \Box\Diamond\varphi$$

where $k \in \mathbb{R}_{\max}$ and $p \in AP$.

- $wLTL\left(AP, \mathbb{R}_{\max}\right)$ the class of all formulas of wLTL over AP and $\mathbb{R}_{\max}$.

wLTL - d-semantics

$0 \leq d < 1$ a *discounting parameter*

Definition

Let $\varphi \in wLTL\left(AP, \mathbb{R}_{\max}\right)$. The infinitary d-semantics of φ is the series

$$\|\varphi\|_d : \left(2^{AP}\right)^{\omega} \rightarrow \mathbb{R}_{\max}$$

For every $w = a_0 a_1 \ldots \in \left(2^{AP}\right)^{\omega}$ we define $\left(\|\varphi\|_d, w\right)$ inductively by:

- $\left(\|k\|_d, w\right) = k$

wLTL - *d*-semantics

$0 \leq d < 1$ a *discounting parameter*

Definition

Let $\varphi \in wLTL\left(AP, \mathbb{R}_{\max}\right)$. The infinitary d-semantics of φ is the series

$$\|\varphi\|_d : \left(2^{AP}\right)^\omega \to \mathbb{R}_{\max}$$

For every $w = a_0 a_1 \ldots \in \left(2^{AP}\right)^\omega$ we define $\left(\|\varphi\|_d, w\right)$ inductively by:

- $\left(\|k\|_d, w\right) = k$
- $\left(\|p\|_d, w\right) = \begin{cases} 0 & \text{if } p \in a_0 \\ -\infty & \text{otherwise} \end{cases}$

$0 \leq d < 1$ a *discounting parameter*

Definition

Let $\varphi \in wLTL\left(AP, \mathbb{R}_{\max}\right)$. The infinitary d-semantics of φ is the series

$$\|\varphi\|_d : \left(2^{AP}\right)^{\omega} \to \mathbb{R}_{\max}$$

For every $w = a_0 a_1 \ldots \in \left(2^{AP}\right)^{\omega}$ we define $\left(\|\varphi\|_d, w\right)$ inductively by:

- $\left(\|k\|_d, w\right) = k$
- $\left(\|p\|_d, w\right) = \begin{cases} 0 & \text{if } p \in a_0 \\ -\infty & \text{otherwise} \end{cases}$
- $\left(\|\neg p\|_d, w\right) = \begin{cases} 0 & \text{if } p \notin a_0 \\ -\infty & \text{otherwise} \end{cases}$

wLTL - *d*-semantics

$0 \leq d < 1$ a *discounting parameter*

Definition

Let $\varphi \in wLTL\,(AP, \mathbb{R}_{\max})$. The infinitary d-semantics of φ is the series

$$\|\varphi\|_d : \left(2^{AP}\right)^{\omega} \to \mathbb{R}_{\max}$$

For every $w = a_0 a_1 \ldots \in \left(2^{AP}\right)^{\omega}$ we define $(\|\varphi\|_d\,, w)$ inductively by:

- $(\|k\|_d\,, w) = k$
- $(\|p\|_d\,, w) = \begin{cases} 0 & \text{if } p \in a_0 \\ -\infty & \text{otherwise} \end{cases}$
- $(\|\neg p\|_d\,, w) = \begin{cases} 0 & \text{if } p \notin a_0 \\ -\infty & \text{otherwise} \end{cases}$
- $(\|\varphi \vee \psi\|_d\,, w) = \max\left((\|\varphi\|_d\,, w), (\|\psi\|_d\,, w)\right)$

wLTL - *d*-semantics

Definition (continued)

- $(\|\varphi \wedge \psi\|_d, w) = (\|\varphi\|_d, w) + (\|\psi\|_d, w)$

wLTL - d-semantics

Definition (continued)

- $(\|\varphi \wedge \psi\|_d, w) = (\|\varphi\|_d, w) + (\|\psi\|_d, w)$
- $(\|\bigcirc\varphi\|_d, w) = d \cdot (\|\varphi\|_d, a_1 a_2 \ldots)$

wLTL - d-semantics

Definition (continued)

- $(\|\varphi \wedge \psi\|_d, w) = (\|\varphi\|_d, w) + (\|\psi\|_d, w)$
- $(\|\bigcirc\varphi\|_d, w) = d \cdot (\|\varphi\|_d, a_1 a_2 \ldots)$
- $(\|\varphi U \psi\|_d, w) =$
  $$\sup_{i \geq 0} \left(\left(\sum_{0 \leq j < i} d^j \cdot (\|\varphi\|_d, a_j a_{j+1} \ldots) + d^i \cdot (\|\psi\|_d, a_i a_{i+1} \ldots) \right) \right)$$

wLTL - *d*-semantics

Definition (continued)

- $(\|\varphi \wedge \psi\|_d, w) = (\|\varphi\|_d, w) + (\|\psi\|_d, w)$
- $(\|\bigcirc\varphi\|_d, w) = d \cdot (\|\varphi\|_d, a_1 a_2 \ldots)$
- $(\|\varphi U \psi\|_d, w) =$
  $$\sup_{i \geq 0} \left(\left(\sum_{0 \leq j < i} d^j \cdot (\|\varphi\|_d, a_j a_{j+1} \ldots) + d^i \cdot (\|\psi\|_d, a_i a_{i+1} \ldots) \right) \right)$$
- $(\|\Box\varphi\|_d, w) = \sum_{i \geq 0} d^i \cdot (\|\varphi\|_d, a_i a_{i+1} \ldots)$

wLTL - *d*-semantics

Definition (continued)

- $(\|\varphi \wedge \psi\|_d, w) = (\|\varphi\|_d, w) + (\|\psi\|_d, w)$
- $(\|\bigcirc\varphi\|_d, w) = d \cdot (\|\varphi\|_d, a_1 a_2 \ldots)$
- $(\|\varphi U \psi\|_d, w) =$
  $$\sup_{i \geq 0}\left(\left(\sum_{0 \leq j < i} d^j \cdot (\|\varphi\|_d, a_j a_{j+1} \ldots) + d^i \cdot (\|\psi\|_d, a_i a_{i+1} \ldots)\right)\right)$$
- $(\|\Box\varphi\|_d, w) = \sum_{i \geq 0} d^i \cdot (\|\varphi\|_d, a_i a_{i+1} \ldots)$
- $(\|\Diamond\varphi\|_d, w) = \sup_{i \geq 0}((\|\varphi\|_d, a_i a_{i+1} \ldots))$

wLTL - d-semantics

Definition (continued)

- $(\|\varphi \wedge \psi\|_d, w) = (\|\varphi\|_d, w) + (\|\psi\|_d, w)$
- $(\|\bigcirc\varphi\|_d, w) = d \cdot (\|\varphi\|_d, a_1 a_2 \ldots)$
- $(\|\varphi U \psi\|_d, w) =$
  $$\sup_{i\geq 0}\left(\left(\sum_{0\leq j<i} d^j \cdot (\|\varphi\|_d, a_j a_{j+1}\ldots) + d^i \cdot (\|\psi\|_d, a_i a_{i+1}\ldots)\right)\right)$$
- $(\|\Box\varphi\|_d, w) = \sum_{i\geq 0} d^i \cdot (\|\varphi\|_d, a_i a_{i+1}\ldots)$
- $(\|\Diamond\varphi\|_d, w) = \sup_{i\geq 0}((\|\varphi\|_d, a_i a_{i+1}\ldots))$
- $(\|\Box\Diamond\varphi\|_d, w) = \sum_{i\geq 0} d^i \cdot \left(\sup_{k\geq i}((\|\varphi\|_d, a_k a_{k+1}\ldots))\right)$

LTL *d*-definability and *d*-recognizability

- An infinitary series $s : (2^{AP})^{\omega} \longrightarrow \mathbb{R}_{\max}$ is called *wLTL-d-definable* if there is a wLTL-formula φ over AP and $\mathbb{R}_{\max}$ such that $s = \|\varphi\|_d$

LTL d-definability and d-recognizability

- An infinitary series $s : (2^{AP})^\omega \to \mathbb{R}_{\max}$ is called *wLTL-d-definable* if there is a wLTL-formula φ over AP and $\mathbb{R}_{\max}$ such that $s = \|\varphi\|_d$
- $\omega\text{-}Ltl(2^{AP}, \mathbb{R}_{\max}, d)$: the class of all wLTL-$d$-definable infinitary series

LTL d-definability and d-recognizability

- An infinitary series $s : (2^{AP})^\omega \to \mathbb{R}_{\max}$ is called *wLTL-d-definable* if there is a wLTL-formula φ over AP and $\mathbb{R}_{\max}$ such that $s = \|\varphi\|_d$
- $\omega\text{-}Ltl(2^{AP}, \mathbb{R}_{\max}, d)$: the class of all wLTL-$d$-definable infinitary series

Theorem (Mandrali 2010)

a fragment of $\omega\text{-}Ltl(2^{AP}, \mathbb{R}_{\max}, d) \subseteq \omega\text{-}Rec(2^{AP}, \mathbb{R}_{\max}, d)$.

LTL d-definability and d-recognizability

- An infinitary series $s : (2^{AP})^{\omega} \rightarrow \mathbb{R}_{\max}$ is called *wLTL-d-definable* if there is a wLTL-formula φ over AP and $\mathbb{R}_{\max}$ such that $s = \|\varphi\|_d$
- $\omega\text{-}Ltl(2^{AP}, \mathbb{R}_{\max}, d)$: the class of all wLTL-$d$-definable infinitary series

Theorem (Mandrali 2010)

$$a \text{ fragment of } \omega\text{-}Ltl(2^{AP}, \mathbb{R}_{\max}, d) \subseteq \omega\text{-}Rec(2^{AP}, \mathbb{R}_{\max}, d).$$

- Open:

LTL d-definability and d-recognizability

- An infinitary series $s : (2^{AP})^\omega \to \mathbb{R}_{\max}$ is called *wLTL-d-definable* if there is a wLTL-formula φ over AP and $\mathbb{R}_{\max}$ such that $s = \|\varphi\|_d$
- ω-$Ltl(2^{AP}, \mathbb{R}_{\max}, d)$: the class of all wLTL-$d$-definable infinitary series

Theorem (Mandrali 2010)

a fragment of ω-$Ltl(2^{AP}, \mathbb{R}_{\max}, d) \subseteq \omega$-$Rec(2^{AP}, \mathbb{R}_{\max}, d)$.

- Open:
  - Is the above inclusion proper? (guess: Yes)

LTL d-definability and d-recognizability

- An infinitary series $s : (2^{AP})^\omega \to \mathbb{R}_{\max}$ is called *wLTL-d-definable* if there is a wLTL-formula φ over AP and $\mathbb{R}_{\max}$ such that $s = \|\varphi\|_d$
- ω-$Ltl(2^{AP}, \mathbb{R}_{\max}, d)$: the class of all wLTL-$d$-definable infinitary series

Theorem (Mandrali 2010)

a fragment of ω-$Ltl(2^{AP}, \mathbb{R}_{\max}, d) \subseteq \omega$-$Rec(2^{AP}, \mathbb{R}_{\max}, d)$.

- Open:
  - Is the above inclusion proper? (guess: Yes)
  - Is the inclusion ω-$Ltl(2^{AP}, \mathbb{R}_{\max}, d) \subseteq \omega$-$Rec(2^{AP}, \mathbb{R}_{\max}, d)$ proper?

Future Work

- Star-free series
- Counter-free weighted automata
- Weighted Monadic First Order logic
- Weighted LTL with past operators
- Decidability results
- Complexity results
- Weighted PSL?
- ...
- Application to Quantitative Model Checking

References

Unweighted setup

- J.R. Büchi, Weak second-order arithmetic and finite automata, Z. Math. Log. Grundl. Math. 6 (1960) 66–92.
- J.R. Büchi, On a decision method in restricted second order arithmetic, in: Proc. 1960 Int. Congr. for Logic, Methodology and Philosophy of Science, 1962, pp. 1–11.
- C. Elgot, Decision problems of finite automata design and related arithmetics, Trans. Amer. Math. Soc. 98 (1961) 21–52.
- M.Y. Vardi and P. Wolper. Reasoning about infinite computations. Information and Computation, 115(1994) 1–37.
- U. Zimmermann, Combinatorial Optimization in Ordered Algebraic Structures, in: Annals of Discrete Mathematics, vol. 10, North-Holland, Amsterdam, 1981.

Weighted automata

- M. Schützenberger, On the definition of a family of automata, Inf. Control 4 (1961) 245–270.

References

Discounting

- L. de Alfaro, T.A. Henzinger, R. Majumdar, Discounting the future in systems theory, in: *Proceedings of ICALP 2003, LNCS* 2719(2003) 1022–1037.
- L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, M. Stoelinga, Model checking discounted temporal properties, *Theoret. Comput. Sci.* 345(2005) 139–170.
- M. Faella, A. Legay, M. Stoelinga, Model checking quantitative linear time logic, *Electron. Notes Theor. Comput. Sci.* 220(2008) 61-77.

Weighted MSO logic

- M. Droste, P. Gastin, Weighted automata and weighted logics, *Theoret. Comput. Sci.* 380(2007) 69-86; extended abstract in: *32nd ICALP*, *LNCS* 3580(2005) 513–525.
- M. Droste, G. Rahonis, Weighted automata and weighted logics on infinite words, *Russian Mathematics (Iz. VUZ)*, 54(1) (2010) 26–45; extended abstract in: *LNCS* 4036(2006) 49–58.

References

Weighted MSO logic with discounting

- M. Droste, G. Rahonis, Weighted automata and weighted logics with discounting, *Theoret. Comput. Sci.* 410(2009) 3481-3494; extended abstract in: *Proceedings of CIAA 2007*, *LNCS* 4783(2007) 73-84.

Weighted LTL with discounting

- M. Mandrdali, Weighted LTL with discounting, preprint presented at *WATA 2010*.

Thank you

Semirings with infinite sums and products

- K is equipped with infinitary sum operations $\sum_I : K^I \to K$, for any index set I, such that for all I and all families $(a_i \mid i \in I)$ of elements of K such that
  - $\sum_{i\in\emptyset} a_i = 0, \quad \sum_{i\in\{j\}} a_i = a_j, \quad \sum_{i\in\{j,k\}} a_i = a_j + a_k$ for $j \neq k$,
  - $\sum_{j\in J}\left(\sum_{i\in I_j} a_i\right) = \sum_{i\in I} a_i$, if $\bigcup_{j\in J} I_j = I$ and $I_j \cap I_{j'} = \emptyset$ for $j \neq j'$,,
  - $\sum_{i\in I}(c \cdot a_i) = c \cdot \left(\sum_{i\in I} a_i\right), \quad \sum_{i\in I}(a_i \cdot c) = \left(\sum_{i\in I} a_i\right) \cdot c,$
- and
- K is endowed with a countably infinite product operation satisfying for all sequences $(a_i \mid i \geq 0)$ of elements of K the following conditions:
  - $\prod_{i\geq 0} 1 = 1, \quad \prod_{i\geq 0} a_i = \prod_{i\geq 0} a_i',$
  - $a_0 \cdot \prod_{i\geq 0} a_{i+1} = \prod_{i\geq 0} a_i, \quad \prod_{j\geq 1} \sum_{i\in I_j} a_i = \sum_{(i_1,i_2,\ldots)\in I_1\times I_2\times\ldots} \prod_{j\geq 1} a_{i_j},$
  - $\prod_{i\geq 0}(a_i \cdot b_i) = \left(\prod\limits_{i\geq 0} a_i\right) \cdot \left(\prod_{i\geq 0} b_i\right)$

  where in the second equation

  $a_0' = a_0 \cdot \ldots \cdot a_{n_1}, a_2' = a_{n_1+1} \cdot \ldots \cdot a_{n_2}, \ldots$ for an increasing sequence