Minule byly

- Konečné automaty (které veškerou informaci udržují ve stavech),
- zásobníkové automaty (kromě stavů mohou použít zásobník),
- poznámky o determinismu a nedeterminismu, tedy
- že deterministický automat je ten, který má přechodovou funkci jednoznačnou,
- že nedeterminismus může pomoci (zásobníkovým automatům), nebo ne (konečným automatům).
- Motivací bylo v prvním přiblížení hledání v textu, ve druhém kroku si naznačíme, co se děje uvnitř překladače, ve třetím kroku si povíme o teoretické informatice.
- Dnes budou Turingovy stroje, gramatiky a souvislosti.

Turingův stroj I

- Oproti konečnému automatu smí na pásku zapisovat a nemusí posunovat hlavu jen doprava (ale i doleva, nebo ji může nechat na místě).

Turingův stroj I

- Oproti konečnému automatu smí na pásku zapisovat a nemusí posunovat hlavu jen doprava (ale i doleva, nebo ji může nechat na místě).
- Existují různé varianty, například s read-only vstupní páskou a read-write pracovní páskou.

Turingův stroj I

- Oproti konečnému automatu smí na pásku zapisovat a nemusí posunovat hlavu jen doprava (ale i doleva, nebo ji může nechat na místě).
- Existují různé varianty, například s read-only vstupní páskou a read-write pracovní páskou.
- Jakýkoliv Turingův stroj disponuje výpočetní silou počítače bez omezené paměti.

Turingův stroj I

- Oproti konečnému automatu smí na pásku zapisovat a nemusí posunovat hlavu jen doprava (ale i doleva, nebo ji může nechat na místě).
- Existují různé varianty, například s read-only vstupní páskou a read-write pracovní páskou.
- Jakýkoliv Turingův stroj disponuje výpočetní silou počítače bez omezené paměti.
- Proč na počtu pásek formálně nezáleží? Kvůli Větě o redukci počtu pásek.

Turingův stroj I

- Oproti konečnému automatu smí na pásku zapisovat a nemusí posunovat hlavu jen doprava (ale i doleva, nebo ji může nechat na místě).
- Existují různé varianty, například s read-only vstupní páskou a read-write pracovní páskou.
- Jakýkoliv Turingův stroj disponuje výpočetní silou počítače bez omezené paměti.
- Proč na počtu pásek formálně nezáleží? Kvůli Větě o redukci počtu pásek.
- Pásku můžeme eliminovat, ovšem za cenu zhoršení složitosti výpočtu.

Turingův stroj II

Definition

Turingovým strojem nazveme strukturu (S, X, F, z, K), kde vše je jako u konečného automatu mimo přechodovou funkci F. Ta je definována takto: $F : S \times X \to X \times \{l, 0, r\} \times S$. Na základě současného stavu a symbolu na pásce přepíšeme obsah pásky, určíme směr pohybu hlavy a přejdeme do nového stavu.

- Poznámky:

Turingův stroj II

Definition

Turingovým strojem nazveme strukturu (S, X, F, z, K), kde vše je jako u konečného automatu mimo přechodovou funkci F. Ta je definována takto: $F : S \times X \to X \times \{l, 0, r\} \times S$. Na základě současného stavu a symbolu na pásce přepíšeme obsah pásky, určíme směr pohybu hlavy a přejdeme do nového stavu.

- Poznámky:
- Občas se Turingovy stroje programují v podobě pětic tvaru: *pablq*.

Turingův stroj II

Definition

Turingovým strojem nazveme strukturu (S, X, F, z, K), kde vše je jako u konečného automatu mimo přechodovou funkci F. Ta je definována takto: $F : S \times X \to X \times \{l, 0, r\} \times S$. Na základě současného stavu a symbolu na pásce přepíšeme obsah pásky, určíme směr pohybu hlavy a přejdeme do nového stavu.

- Poznámky:
- Občas se Turingovy stroje programují v podobě pětic tvaru: *pablq*.
- Opět má smysl uvažovat o deterministickém (DTS) a nedeterministickém Turingově stroji (NTS).

Turingův stroj II

Definition

Turingovým strojem nazveme strukturu (S, X, F, z, K), kde vše je jako u konečného automatu mimo přechodovou funkci F. Ta je definována takto: $F : S \times X \to X \times \{l, 0, r\} \times S$. Na základě současného stavu a symbolu na pásce přepíšeme obsah pásky, určíme směr pohybu hlavy a přejdeme do nového stavu.

- Poznámky:
- Občas se Turingovy stroje programují v podobě pětic tvaru: *pablq*.
- Opět má smysl uvažovat o deterministickém (DTS) a nedeterministickém Turingově stroji (NTS).
- Kontrolní otázka: Myslíte, že DTS je stejně výkonný jako NTS?

Turingův stroj III

Je DTS a NTS stejně výkonný:

- Z hlediska vyčíslitelnosti?

Turingův stroj III

Je DTS a NTS stejně výkonný:

- Z hlediska vyčíslitelnosti?
- Z hlediska složitosti?

Turingův stroj III

Je DTS a NTS stejně výkonný:

- Z hlediska vyčíslitelnosti?
- Z hlediska složitosti?
- Z hlediska zachování polynomiální složitosti?

Turingův stroj III

Je DTS a NTS stejně výkonný:

- Z hlediska vyčíslitelnosti?
- Z hlediska složitosti?
- Z hlediska zachování polynomiální složitosti?
- Poslední problém je jedním z Problémů milénia (zvaný P- a NP-hypotéza).

Gramatiky I

- Automaty jsme používali k rozpoznání množiny slov (zpravidla nekonečné) zvané jazyk.

Gramatiky I

- Automaty jsme používali k rozpoznání množiny slov (zpravidla nekonečné) zvané jazyk.
- Kromě automatů můžeme použít i jiný abstraktní způsob popisu jazyků a to tak, že konečnou množinou pravidel popíšeme, jak všechna taková slova vypadají.

Gramatiky I

- Automaty jsme používali k rozpoznání množiny slov (zpravidla nekonečné) zvané jazyk.
- Kromě automatů můžeme použít i jiný abstraktní způsob popisu jazyků a to tak, že konečnou množinou pravidel popíšeme, jak všechna taková slova vypadají.
- Automaty jsou snadné na pochopení, ale obtížné k pořádnému sestrojení, proto potřebujeme i formalismus snáze použitelný, i když působí abstraktnějším dojmem.

Gramatiky I

- Automaty jsme používali k rozpoznání množiny slov (zpravidla nekonečné) zvané jazyk.
- Kromě automatů můžeme použít i jiný abstraktní způsob popisu jazyků a to tak, že konečnou množinou pravidel popíšeme, jak všechna taková slova vypadají.
- Automaty jsou snadné na pochopení, ale obtížné k pořádnému sestrojení, proto potřebujeme i formalismus snáze použitelný, i když působí abstraktnějším dojmem.
- Nasadíme proto gramatiky.

Gramatiky I

- Automaty jsme používali k rozpoznání množiny slov (zpravidla nekonečné) zvané jazyk.
- Kromě automatů můžeme použít i jiný abstraktní způsob popisu jazyků a to tak, že konečnou množinou pravidel popíšeme, jak všechna taková slova vypadají.
- Automaty jsou snadné na pochopení, ale obtížné k pořádnému sestrojení, proto potřebujeme i formalismus snáze použitelný, i když působí abstraktnějším dojmem.
- Nasadíme proto gramatiky.
- Anglická gramatika: $W \longrightarrow SVOMPTW$,

Gramatiky I

- Automaty jsme používali k rozpoznání množiny slov (zpravidla nekonečné) zvané jazyk.
- Kromě automatů můžeme použít i jiný abstraktní způsob popisu jazyků a to tak, že konečnou množinou pravidel popíšeme, jak všechna taková slova vypadají.
- Automaty jsou snadné na pochopení, ale obtížné k pořádnému sestrojení, proto potřebujeme i formalismus snáze použitelný, i když působí abstraktnějším dojmem.
- Nasadíme proto gramatiky.
- Anglická gramatika: $W \longrightarrow SVOMPTW$,
- A také $W \longrightarrow SVOMPT$.

Gramatiky II

- Napsat gramatiku přirozeného jazyka je prakticky vyloučené (informatici to zkoušeli),

Gramatiky II

- Napsat gramatiku přirozeného jazyka je prakticky vyloučené (informatici to zkoušeli),
- dnešní trendy ve výpočetní lingvistice od gramatik ustupují (ačkoliv tzv. pravidlová lingvistika je stále významným směrem), nasazují se statistické metody.

Gramatiky II

- Napsat gramatiku přirozeného jazyka je prakticky vyloučené (informatici to zkoušeli),
- dnešní trendy ve výpočetní lingvistice od gramatik ustupují (ačkoliv tzv. pravidlová lingvistika je stále významným směrem), nasazují se statistické metody.
- Lingvisty vyvinutý aparát se ale ohromně osvědčil na strojové (formální) jazyky.

Gramatiky III

Definition

Gramatikou nazveme trojici (T, N, P, S), kde T je konečná množina terminálů (znaků, z nichž budou výsledná slova sestávat), N je konečná množina tzv. neterminálů (pomocných symbolů), S je startovní neterminál a P je konečná množina tzv. přepisovacích pravidel. Každé přepisovací pravidlo lze zapsat ve tvaru $(T \cup N)* \rightarrow (T \cup N \cup \lambda)*$, tedy posloupnost terminálů a neterminálů přepíšeme také na posloupnost terminálů a neterminálů, ovšem **terminály musejí zůstat zachovány.**

Gramatiky IV

- Terminály obvykle označujeme malými písmeny, neterminály velkými.

Gramatiky IV

- Terminály obvykle označujeme malými písmeny, neterminály velkými.
- Příklady: Gramatika pro jazyk $a * b*$?

Gramatiky IV

- Terminály obvykle označujeme malými písmeny, neterminály velkými.
- Příklady: Gramatika pro jazyk $a * b*$?
- $(\{a, b\}, \{S, T\}, P, S)$, kde
  $P = \{S \to aS, S \to T, T \to bT, T \to \lambda\}$.

Gramatiky IV

- Terminály obvykle označujeme malými písmeny, neterminály velkými.
- Příklady: Gramatika pro jazyk $a * b*$?
- $(\{a, b\}, \{S, T\}, P, S)$, kde
  $P = \{S \to aS, S \to T, T \to bT, T \to \lambda\}$.
- Gramatika pro jazyk $a^n b^n$?

Gramatiky IV

- Terminály obvykle označujeme malými písmeny, neterminály velkými.
- Příklady: Gramatika pro jazyk $a * b*$?
- $(\{a, b\}, \{S, T\}, P, S)$, kde
  $P = \{S \to aS, S \to T, T \to bT, T \to \lambda\}$.
- Gramatika pro jazyk $a^n b^n$?
- $(\{a, b\}, \{S\}, P, S)$, kde $P = \{S \to \lambda, S \to aSb\}$.

Gramatiky IV

- Terminály obvykle označujeme malými písmeny, neterminály velkými.
- Příklady: Gramatika pro jazyk $a * b*$?
- $(\{a, b\}, \{S, T\}, P, S)$, kde
  $P = \{S \to aS, S \to T, T \to bT, T \to \lambda\}$.
- Gramatika pro jazyk $a^n b^n$?
- $(\{a, b\}, \{S\}, P, S)$, kde $P = \{S \to \lambda, S \to aSb\}$.
- Gramatika pro čísla dělitelná dvěma?

Gramatiky IV

- Terminály obvykle označujeme malými písmeny, neterminály velkými.
- Příklady: Gramatika pro jazyk $a * b*$?
- $(\{a, b\}, \{S, T\}, P, S)$, kde
  $P = \{S \to aS, S \to T, T \to bT, T \to \lambda\}$.
- Gramatika pro jazyk $a^n b^n$?
- $(\{a, b\}, \{S\}, P, S)$, kde $P = \{S \to \lambda, S \to aSb\}$.
- Gramatika pro čísla dělitelná dvěma?
- $(\{0 - 9\}, \{S\}, P, S)$, kde $P = \{S \to [02468], S \to [0 - 9\S\}$

Gramatiky IV

- Terminály obvykle označujeme malými písmeny, neterminály velkými.
- Příklady: Gramatika pro jazyk $a * b*$?
- $(\{a, b\}, \{S, T\}, P, S)$, kde
  $P = \{S \to aS, S \to T, T \to bT, T \to \lambda\}$.
- Gramatika pro jazyk $a^n b^n$?
- $(\{a, b\}, \{S\}, P, S)$, kde $P = \{S \to \lambda, S \to aSb\}$.
- Gramatika pro čísla dělitelná dvěma?
- $(\{0 - 9\}, \{S\}, P, S)$, kde $P = \{S \to [02468], S \to [0 - 9\S\}$
- Cvičení: Gramatika pro čísla dělitelná třemi a šesti.

Gramatiky IV

- Terminály obvykle označujeme malými písmeny, neterminály velkými.
- Příklady: Gramatika pro jazyk $a * b*$?
- $(\{a, b\}, \{S, T\}, P, S)$, kde $P = \{S \to aS, S \to T, T \to bT, T \to \lambda\}$.
- Gramatika pro jazyk $a^n b^n$?
- $(\{a, b\}, \{S\}, P, S)$, kde $P = \{S \to \lambda, S \to aSb\}$.
- Gramatika pro čísla dělitelná dvěma?
- $(\{0 - 9\}, \{S\}, P, S)$, kde $P = \{S \to [02468], S \to [0 - 9\S\}$
- Cvičení: Gramatika pro čísla dělitelná třemi a šesti.
- Cvičení*: Gramatika Pascalu?

Souvislosti automatů a gramatik

- Automaty a gramatiky dělají totéž, jak by tedy mohly vypadat gramatické ekvivalenty jednotlivých automatů?

Souvislosti automatů a gramatik

- Automaty a gramatiky dělají totéž, jak by tedy mohly vypadat gramatické ekvivalenty jednotlivých automatů?
- Gramatiku nazveme regulární, pokud všechna její pravidla jsou tvaru $S \longrightarrow tN$, kde S a N jsou dva (ne nutně různé) neterminály a t je posloupnost terminálů.

Souvislosti automatů a gramatik

- Automaty a gramatiky dělají totéž, jak by tedy mohly vypadat gramatické ekvivalenty jednotlivých automatů?
- Gramatiku nazveme regulární, pokud všechna její pravidla jsou tvaru $S \to tN$, kde S a N jsou dva (ne nutně různé) neterminály a t je posloupnost terminálů.
- Gramatiku nazveme bezkontextovou, pokud všechna její pravidla jsou tvaru $S \to N$, kde S je neterminál a N je posloupnost terminálů a neterminálů.

Souvislosti automatů a gramatik

- Automaty a gramatiky dělají totéž, jak by tedy mohly vypadat gramatické ekvivalenty jednotlivých automatů?
- Gramatiku nazveme regulární, pokud všechna její pravidla jsou tvaru $S \rightarrow tN$, kde S a N jsou dva (ne nutně různé) neterminály a t je posloupnost terminálů.
- Gramatiku nazveme bezkontextovou, pokud všechna její pravidla jsou tvaru $S \rightarrow N$, kde S je neterminál a N je posloupnost terminálů a neterminálů.
- Gramatiku nazveme nezkracující, pokud žádný neterminál nepřepíšeme na prázdný symbol.

Souvislosti automatů a gramatik

- Automaty a gramatiky dělají totéž, jak by tedy mohly vypadat gramatické ekvivalenty jednotlivých automatů?
- Gramatiku nazveme regulární, pokud všechna její pravidla jsou tvaru $S \to tN$, kde S a N jsou dva (ne nutně různé) neterminály a t je posloupnost terminálů.
- Gramatiku nazveme bezkontextovou, pokud všechna její pravidla jsou tvaru $S \to N$, kde S je neterminál a N je posloupnost terminálů a neterminálů.
- Gramatiku nazveme nezkracující, pokud žádný neterminál nepřepíšeme na prázdný symbol.
- Jazyky generované příslušnými gramatikami nazveme regulární, resp. bezkontextové (nezkracující jazyky jsem ještě neslyšel).

Kleenova věta

Theorem

- *Regulární jazyky jsou rozpoznatelné konečnými automaty (a žádné jiné jazyky konečný automat rozpoznat neumí).*
- *Bezkontextové jazyky jsou rozpoznatelné nedeterministickými zásobníkovými automaty (a naopak žádné jiné jazyky nedeterministický zásobníkový automat rozpoznat neumí).*

Důkaz (náznak).

Náznak pro regulární jazyky: Neterminály odpovídají stavům.

□

Kleenova věta

Theorem

- *Regulární jazyky jsou rozpoznatelné konečnými automaty (a žádné jiné jazyky konečný automat rozpoznat neumí).*
- *Bezkontextové jazyky jsou rozpoznatelné nedeterministickými zásobníkovými automaty (a naopak žádné jiné jazyky nedeterministický zásobníkový automat rozpoznat neumí).*

Důkaz (náznak).

Náznak pro bezkontextové jazyky: Nezpracované neterminály (a čekající terminály) strkáme ve správném pořadí na zásobník a zpracováváme. Z automatu pak uděláme pravidlo tak, že jedno pravidlo má území platnosti od uložení konkrétního symbolu na zásobník po jeho vyjmutí (popř. zpracování).

Kleenova věta

Theorem

- *Regulární jazyky jsou rozpoznatelné konečnými automaty (a žádné jiné jazyky konečný automat rozpoznat neumí).*
- *Bezkontextové jazyky jsou rozpoznatelné nedeterministickými zásobníkovými automaty (a naopak žádné jiné jazyky nedeterministický zásobníkový automat rozpoznat neumí).*

Theorem

Regulární jazyky jsou právě jazyky popsatelné regulárními výrazy.

Zpřátelené a znepřátelené automaty

aneb zpátky na stromy, tedy k automatům

- Konečný automat se dobře implementuje, ale rozpozná velmi omezené jazyky.

Zpřátelené a znepřátelené automaty

aneb zpátky na stromy, tedy k automatům

- Konečný automat se dobře implementuje, ale rozpozná velmi omezené jazyky.
- Deterministický zásobníkový automat také.

Zpřátelené a znepřátelené automaty

aneb zpátky na stromy, tedy k automatům

- Konečný automat se dobře implementuje, ale rozpozná velmi omezené jazyky.
- Deterministický zásobníkový automat také.
- Turingův stroj má výpočetní sílu počítače s neomezenou pamětí (tedy jakýkoliv program lze přeložit pro Turingův stroj) a umí tedy rozpoznat všechny algoritmicky rozpoznatelné jazyky, s jeho simulací to ale není žádná sláva (může se zacyklit).

Znepřátelené automaty

jak je ochočit...

- Nedeterministický zásobníkový automat je na tom podobně jako Turingův stroj (nerozpozná všechny jazyky, zacyklení dovedeme poznat snáze, ale potíž je s nedeterminismem).

Znepřátelené automaty

jak je ochočit...

- Nedeterministický zásobníkový automat je na tom podobně jako Turingův stroj (nerozpozná všechny jazyky, zacyklení dovedeme poznat snáze, ale potíž je s nedeterminismem).
- Nedeterministické zásobníkové automaty ovšem bezpečně postačují pro rozpoznání strojových jazyků (deterministické zásobníkové automaty mají problém).

Znepřátelené automaty

jak je ochočit...

- Nedeterministický zásobníkový automat je na tom podobně jako Turingův stroj (nerozpozná všechny jazyky, zacyklení dovedeme poznat snáze, ale potíž je s nedeterminismem).
- Nedeterministické zásobníkové automaty ovšem bezpečně postačují pro rozpoznání strojových jazyků (deterministické zásobníkové automaty mají problém).
- Proto se definují automaty rozpoznávající alespoň všechny regulární jazyky a některé (jednodušší) bezkontextové.

Jak vypadá překladač zevnitř?

Při pohledu z velké dálky uvidíme toto...

- Řekli jsme si, že bychom rozbili vstup na jednotlivé vstupní prvky (zvané tokeny) a pak se jakýmsi nedeterministickým postupem dobereme kódu.

Jak vypadá překladač zevnitř?

Při pohledu z velké dálky uvidíme toto...

- Řekli jsme si, že bychom rozbili vstup na jednotlivé vstupní prvky (zvané tokeny) a pak se jakýmsi nedeterministickým postupem dobereme kódu.
- Všimneme si ovšem, že programovací jazyk je především formální jazyk a lze ho tudíž popsat gramatikou.

Jak vypadá překladač zevnitř?

Při pohledu z velké dálky uvidíme toto...

- Řekli jsme si, že bychom rozbili vstup na jednotlivé vstupní prvky (zvané tokeny) a pak se jakýmsi nedeterministickým postupem dobereme kódu.
- Všimneme si ovšem, že programovací jazyk je především formální jazyk a lze ho tudíž popsat gramatikou.
- Podle gramatiky lze vypěstovat syntaktický strom, který říká, jak je program (syntakticky) postaven.

Jak vypadá překladač zevnitř?

Při pohledu z velké dálky uvidíme toto...

- Řekli jsme si, že bychom rozbili vstup na jednotlivé vstupní prvky (zvané tokeny) a pak se jakýmsi nedeterministickým postupem dobereme kódu.
- Všimneme si ovšem, že programovací jazyk je především formální jazyk a lze ho tudíž popsat gramatikou.
- Podle gramatiky lze vypěstovat syntaktický strom, který říká, jak je program (syntakticky) postaven.
- Dále domyslíme sémantiku (smysl a význam) jednotlivých syntaktických konstrukcí

Jak vypadá překladač zevnitř?

Při pohledu z velké dálky uvidíme toto...

- Řekli jsme si, že bychom rozbili vstup na jednotlivé vstupní prvky (zvané tokeny) a pak se jakýmsi nedeterministickým postupem dobereme kódu.
- Všimneme si ovšem, že programovací jazyk je především formální jazyk a lze ho tudíž popsat gramatikou.
- Podle gramatiky lze vypěstovat syntaktický strom, který říká, jak je program (syntakticky) postaven.
- Dále domyslíme sémantiku (smysl a význam) jednotlivých syntaktických konstrukcí
- a vygenerujeme kód.

Jak vypadá překladač zevnitř?

Při pohledu z velké dálky uvidíme toto...

- Řekli jsme si, že bychom rozbili vstup na jednotlivé vstupní prvky (zvané tokeny) a pak se jakýmsi nedeterministickým postupem dobereme kódu.
- Všimneme si ovšem, že programovací jazyk je především formální jazyk a lze ho tudíž popsat gramatikou.
- Podle gramatiky lze vypěstovat syntaktický strom, který říká, jak je program (syntakticky) postaven.
- Dále domyslíme sémantiku (smysl a význam) jednotlivých syntaktických konstrukcí
- a vygenerujeme kód.
- Posledním krokem se zabývat nebudeme (je pod naši úroveň a nad naše schopnosti), po konci sémantické analýzy jsme schopni kód zinterpretovat (tedy postavit interpret).

Pohled inženýrský

- Existují automaty, které jsou schopné rozpoznávat regulární výrazy (flex),

Pohled inženýrský

- Existují automaty, které jsou schopné rozpoznávat regulární výrazy (flex),
- existují automaty schopné rozpoznat i některé bezkontextové gramatiky (bison).

Pohled inženýrský

- Existují automaty, které jsou schopné rozpoznávat regulární výrazy (flex),
- existují automaty schopné rozpoznat i některé bezkontextové gramatiky (bison).
- Použijeme tudíž tuto osvědčenou dvojici nástrojů. Prvním rozbijeme vstup na tokeny, druhý provede syntaktickou analýzu (ze které dovedeme postavit syntaktický strom).

Pohled inženýrský

- Existují automaty, které jsou schopné rozpoznávat regulární výrazy (flex),
- existují automaty schopné rozpoznat i některé bezkontextové gramatiky (bison).
- Použijeme tudíž tuto osvědčenou dvojici nástrojů. Prvním rozbijeme vstup na tokeny, druhý provede syntaktickou analýzu (ze které dovedeme postavit syntaktický strom).
- Při troše štěstí (u netypových jazyků) je sémantická analýza prázdná a máme hotovo.

Pohled inženýrský

- Existují automaty, které jsou schopné rozpoznávat regulární výrazy (flex),
- existují automaty schopné rozpoznat i některé bezkontextové gramatiky (bison).
- Použijeme tudíž tuto osvědčenou dvojici nástrojů. Prvním rozbijeme vstup na tokeny, druhý provede syntaktickou analýzu (ze které dovedeme postavit syntaktický strom).
- Při troše štěstí (u netypových jazyků) je sémantická analýza prázdná a máme hotovo.
- Při troše smůly u bisonem nehneme.

Pohled inženýrský

- Existují automaty, které jsou schopné rozpoznávat regulární výrazy (flex),
- existují automaty schopné rozpoznat i některé bezkontextové gramatiky (bison).
- Použijeme tudíž tuto osvědčenou dvojici nástrojů. Prvním rozbijeme vstup na tokeny, druhý provede syntaktickou analýzu (ze které dovedeme postavit syntaktický strom).
- Při troše štěstí (u netypových jazyků) je sémantická analýza prázdná a máme hotovo.
- Při troše smůly u bisonem nehneme.
- Flex bychom byli schopni oběhnout (vlastním automatem), bisona zatím ne.