1 Obecné pojmy

1.1 Grafy

- Neorientovaný graf (V, E) kde $e \in E$ sú dvojprvkové podmnožiny V (neusporiadné).
- Orientovaný graf (V, E) kde $E \subseteq V \times V$.
- Izomorfixmus grafov $G \simeq H$: $f : V(G) \to V(H)$ t.ž. $(u, v) \in E(G)$ iff $(f(u), f(v)) \in E(H)$.
- Planárny (rovinný) graf: Dá sa nakresliť do roviny bez pretínania hrán.
- (Silno) súvislá komponenta — zo všetký vrcholov viem dosiahnuť všetky vrcholy.

1.2 Výroková logika

- Booleovská funkcia $F : [0,1]^n \to [0,1]$. Operátory $\wedge, \vee, \neg, \implies, ...$ sú funkcie.
- Formula v systéme $\mathcal{L}(F_0, ..., F_n)$: $\varphi ::= x \in Var \mid F_i(\varphi_0, ..., \varphi_m)$
- Modelom formule je valuácia literálov $v : Var \to [0,1]$: $v \models \varphi$
- Pravdivá/nepravdivá $v \models \varphi$, splniteľná $\exists v : v \models \varphi$, tautológia $\forall v : v \models \varphi$ ($\models \varphi$)
- φ je tautologický dôsledok: $T \models \varphi$ (bez ohľadu na model)
- Normálny tvar: CNF/DNF (klauzula, duálna klauzula, literál)

1.3 Logika prvého rádu

- Premenné $Var : x, ...$, funkčné symboly $F : f_0, ...$, predikátové symboly $\mathcal{P} : P_0, ...$
- Jazyk: Sada funkčných a predikátových symbolov
- Realizácia $\mathcal{M}$: Univerzum M, relácie $P_i \subseteq M^m$, funkcie $f_i : M^m \to M$
- **S rovnosťou vs. bez rovnosti**
- Term: $t ::= x \in Var \mid f_i(t_0, ..., t_m)$
- Formula $\varphi ::= P_i(t_0, ..., t_m) \mid (t_0 = t_1) \mid \varphi_0 \to \varphi_1 \mid \neg\varphi_0 \mid \exists x : \varphi$
- Modelom formule je realizácia a valuácia $v : Var \to M$: $(\mathcal{M}, v) \models \varphi$
- Pravdivá $\mathcal{M} \models \varphi$ (bez ohľadu na valuáciu)
- φ je sémantický dôsledok: $T \models \varphi$ (bez ohľadu na model)
- Teória T je množina axiómov. Teória má model $\mathcal{M}$ ak sú v nej všetky axiómy T pravdivé.

1.4 Odvodzovacie systémy

- Sada (generických) axiómov a syntaktických odvodzovacích pravidiel.
- Dôkaz: Postupnosť $\varphi_0, \varphi_1, ...$ kde φ_i je axióm, alebo φ_i vznikne pravidlom z $\varphi_j : j < i$.
- Dokázateľná formula: $\vdash \varphi$, Dokázateľná z predpokladu: $T \vdash \varphi$
- Korektnosť: Dokázateľné je pravdivé ($\vdash \varphi \implies \models \varphi$)
- Úplnosť: Pravdivé je dokázateľné ($\models \varphi \implies \vdash \varphi$)
- Sporná = všetky formule sú dokázateľné.

1.5 LTS a Kripkeho štruktúra

- $LTS = (S, A, \Delta)$ — stavy, akcie a prechody $\Delta \subseteq S \times A \times S$
- $K = (S, T, s_0, L)$ — kde $T \subseteq S \times S$, inicálny stav s_0 a proposition labelling $L : S \to 2^{AP}$. Väčšinou chcem T totálnu.
- Prechodový systém — T je množina prechodov kde prechod je deterministická funkcia ($T \subseteq \mathcal{P}(S \to S)$). Interpretujem ako že mám pomenované prechody.

1.6 Trace ekvivalencia a bisimulácia

- Trace: jeden beh v LTS
- Trace ekvivalencia = množiny behov systémov sú rovnaké.
- Relácia R je bisimulácia ak spĺňa, že ak $(s, t) \in R$ a $(s, a, s') \in \Delta_0$, tak musí existovať $(t, a, t') \in \Delta_1$ t.ž $(s', t') \in R$ a to isté opačne.
- Dva stavy sú bisimulačne ekvivalentné ak existuje nejaké bisimulácia v ktorej sú v relácií.
- Relácia bisimilarity $\sim$ je najväčšia ekvivalencia ktorá je súčasne bisimulácia.
- Pozn.: Môžem mať situáciu kedy p simuluje q a q simuluje p, ale rôznymi reláciami a teda neviem zostrojiť jednu bisimuláciu ktorá ich zjednotí (Príklad: jeden proces sa rozhodne neskôr a druhý sa rozhodne buď hneď alebo neskôr.)

1.7 Omega automaty

- Omega regulárne jazyky sú tvaru $U.V^\omega$ kde U a V sú regulárne jazyky.
- Buchi automat $(Q, \Sigma, \delta, s_0, F)$ — stavy, abeceda, prechodová relácia (nedeterministická), inicálny stav, akceptujúce stavy.
- Automat akceptuje ak sa v slove nachádza stav z F nekonečne často.
- Zobecnený Buchiho automat: Miesto F mám množinu množín stavov a akceptujem ak prejdem každou z nich nekonečne často.

1.8 CPO a Fixed point

- Complete lattice (úplný svaz): Každá podmnožina má suprémum aj infimum.
- Complete partial order (CPO): Každá nekonečná postupnosť $a_0 \sqsubseteq a_1 \sqsubseteq ...$ má v množine súprémum.
- Directed complete partial order (DCPO): Každá directed množina má suprémum. Directed množina je taká že po dvoch tam majú prvky upper bound.
- Pozn.: každý complete lattice je aj CPO, každý konečný complete lattice je aj DCPO.
- Veta o pevnom bode (Knaster-Tarski): Pre complete lattice $(L, \leq)$ a monotónnu funkciu $f : L \to L$ platí, že aj fixed pointy f tvoria complete lattice.
- Veta o pevnom bode (Kleene): Pre DCPO $(L, \leq)$ a Scott-spojitú (monotónnu) funkciu $f : L \to L$ existuje najmenší pevný bod ktorý a je suprémom postupnosti $f^n(\bot)$. (Zjednodušenie je pre konečný complete lattice a monotónnu funkciu)

1.9 LTL

- $\varphi ::= true \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid X\ \varphi \mid \varphi\ U\ \psi$
- Iné operátory: $F\varphi = true\ U\varphi$, $G = \neg F \neg\varphi$, $\varphi\ R\ \psi = \neg(\neg\varphi\ U\ \neg\psi)$ a $\varphi\ W\ \psi = (\varphi\ U\ \psi) \vee G\varphi$
- Formula ktorú nevyjadrím v CTL: $F(G(\phi))$

1.10 CTL

- $\varphi ::= true \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid AX\ \varphi \mid E/A[\varphi\ U\ \psi]$
- Iné operátory: $EX\varphi = \neg AX\neg\varphi$, $E/AF\varphi = E/A[true\ U\ \varphi]$, $E/AG\varphi = \neg A/EF\neg\varphi$
- Formula ktorú nevyjadrím v LTL: $AG(EF(\phi))$

1.11 Real Time Systems

- Procesor (aktívny), zdroj (pasívny), job (jednotka práce), task (periodicky sa opakujúci job)
- Release time r, execution time e, relative/absolute deadline d/D, period p, completion time C, response time $C - r$, utilization $\frac{e}{p}$, density tasku $\frac{e}{min(p,D)}$ ($\frac{e}{d-r}$ pre job)
- Hard: verification, Soft: validation
- Tasks: Periodic (hard), Aperiodic (soft), Sporadic (hard)

1.12 Distribuované systémy

- Komunikujúce procesy prepojené sieťou (Iné modely: MapReduce).
- Sieť je buď synchrónna alebo asynchrónna (všeobecnejšie).
- Komunikácia je drahá — Komunikačná zložitosť (počet, res. veľkosť správ)
- Sieť môže mať rôznu topológiu: ring, star, grid, hypercube, torus, tree, complete graph, unknown
- Zlyhania procesov: nečakaný delay, jednorázové zlyhanie, zlyhanie a zotavenie, bizantínske zlyhanie
- Zlyhania siete: nečakaný delay, packet loss, packet duplication, packet reordering, bizantínske zlyhanie

2 Advanced pojmy

2.1 Grafové problémy

- Nezávislá množina: žiadne dva vrcholy v množine nie sú spojené hranou.
- Vrcholové pokrytie: každá hrana končí vo vrchole ktorý je v pokrytí.
- Dominujúca množina: každý vrchol má suseda v množine.
- Ofarbenie grafu: Každý vrchol má medzi susedmi unikátnu farbu.
- Hamiltonovský cyklus/cesta.
- Reprezentácia grafov: Matica susednosti, Incidenčná matica, zoznam nasledníkov (hrán), implicitne.

2.2 Logika

- Úplný systém logických spojok: dá sa pomocou neho vyjadriť ľubovoľná logická funkcia. (Dôkaz pre ano/or/not - ľubovoľnú funkciu zákódujem do veľmi hnusného DNF)
- Shefferovská funkcia: úplná sama o sebe (zhruba 1/3 funkcií je shefferovská)
- Lukasiewicz (odvodzovacý systém pre výrokovú logiku):
  - A1: $A \to (B \to A)$
  - A2: $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$
  - A3: $(\neg A \to \neg B) \to (B \to A)$
  - MP (modus ponens): Z $A \to B$ a A vyvoď B
- First order logic (odvodzovacý systém):
  - A1-A3 + MP
  - $\forall x.\varphi \to \varphi[x/t]$ (špecifikácia)
  - $\forall x.(A \to B) \to (A \to \forall x.B)$ (ak x nemá voľný výskyt v A, distribúcia)
  - Generalizácia: Z A odvoď $\forall x.A$
  - Rovnosť: $x = x$, ak $x = y$, tak $f(x) = f(y)$ a ak $x = y \wedge P(x)$, tak $P(y)$.
- Veta o dedukcií: Ak $T \vdash (A \to B)$ tak $T \cup A \vdash B$.
- Veta o kompaktnosti: Nekonečná teória má model práve keď každá konečná podteória má model.
- Löwenheim-Skolem: Ak pre ľubovolné n existuje model s nosičom mohutnosti n, tak existuje aj nekonečný model.

2.3 Gödel

- Peanova aritmetika — axiomy aritmetiky celých čísel v predikátovej logike.
- Gödelov predikát — $\beta(a, b, i, x) \iff x = a \mod (1 + b(1 + i))$ — kóduje unikátnu (konečnú) postupnosť čísel.
- 1. Veta o neúplnosti: Existuje formula pravdivá v $\mathcal{N}$ ktorá nie je dokázateľná v Peanovej aritmetike.
- Proof sketch: Dôkazy sa dajú kódovať ako čísla, teda môžem mať predikát $proof(x, y)$ ak x je dôkaz y, z tohoto dostanem predikát pre "x je dokázateľné". Na základe toho potom skonštruujem formulu typu $\varphi = \neg provable(\varphi)$. Formula sa nazýva Gödel sentence.
- 2. Veta o neúplnosti: Formula $consis = \neg provable(a \wedge \neg a)$ (ekvivalentné dôkazu bezspornosti) nie je dokázateľná v Peanovej aritmetike pre ľubovoľné a (ani iných podobne silných systémoch).
- Proof sketch: Použijem Gödel sentence g ako a a ukážem že ak $\vdash g$, tak aj $\vdash provable(g)$ a súčasne $\vdash \neg provable(g)$ (toto sa dá dokázať v PA, keďže g je nedokázateľná). Potom ale z $\vdash provable(g)$ plynie $\neg consis$.

2.4 Logiky nad slovami

- Predikátová logika — kvantifikujem nad pozíciami v slove a mám dva predikáty $Q_a(x)$ (na pozícií x je znak a) a $x < y$.
- Pomerne slabá logika, nevie vyjadriť ani jazyk slov párnej dĺžky.
- Second order logic — pridávam kvantifikáciu nad množinami pozícií a predikát $x \in X$.
- Populárne predikáty: $next$, $last$, $first$... Množiny potom väčšinou definujem induktívne (existuje množina taká že jej prvok je prvý v slove a všetky jej prvky sú o dva za iným prvkom)
- FA to MSOL:
  - Definujem si "partície"do ktorých sa mi rozpadnú pozície slova podľa toho či skončím v tom stave keď prečítam danú pozíciu.
  - Inicializácia: Po prečítaní prvého znaku skončím v množine danej týmto znakom.
  - Množiny pozícií sú dizjunktné a každá pozícia patrí do jednej množiny.
  - Pozície zachovávajú prechodovú funkciu.
  - Potom už stačí napísať že posledná pozícia leží v množine ktorá odpovedá akceptujúcemu stavu.
- MSOL to FA:
  - Postupujem induktívne po formuli. Konštruujem automat ktorý akceptuje slová abecedy $\Sigma \times [0,1]^n$ kde n je počet voľných premenných a hovorí či je pozícia v danej premennej alebo nie (FO premenné budú mať v celom slove len jednu jednotku).
  - Preklad predikátov je triviálny (proste sa presuniem keď na razím na vhodné kombo znak-príslušnosť)
  - Negácia — nemôžem robiť hlúpy komplement, musím ešte zabezpečiť že nebudem akceptovať malformed slová (FO premenné s viac jednotkami). Viem ale urobiť jazyk validných slov, takže urobím komplement a sprienikujem s týmto jazykom.
  - Dizjunkcia — rozšírim abecedy jazykov na rovnakú doménu (jeden prechod zdvojím aby šiel bez ohľadu na novú premennú). Potom môžem robiť union.
  - Kvantifikácia — proste useknem celú pozíciu premennej z abecedy.
  - Výsledný automat môže byť veľký — "veža exponentov"vhľadom k počtu negácií.

2.5 Omega automaty

- Uzavretosť na zjednotenie (blbý union), zreťazenie regulárny+omega, res. regulárny$^\omega$
- Prienik: Konštrukcia ako pre prevod zobecneného automatu: Vytvorím vrstvy pre každý prienikovaný automat a behám medzi vrstvami pri prechode cez koncový stav.
- Komplement: Ano, ale dôkaz je fuj **??????**
- Neprázdnosť jazyka je NLOG úplná (Reachability), Ne-univerzálnosť je PSPACE.
- Muller — množiny stavov, akceptujem ak $inf(w)$ presne matchuje jednu z množín.
- Rabin — páry množín, akcetujem ak existujé pár t.ž. $inf(w)$ má prienik s prvou ale nie s druhou.
- Street — páry množín, akceptujem ak pre všetky páry $inf(w)$ má prienik s prvou implikuje že má prienik s druhou.

2.6 Sémantiky: program

- Program je reprezentovaný syntaktickým stromom (atomy majú doménu a môžu mať potenciálne svoj vlastný AST, ale to ma až tak nezaujíma)
- Domény: Var, $Bool$ a Num
- Aritmetické výrazy $a ::= Num \mid Var \mid a + a \mid a - a \mid a \cdot a$
- Booleovské výrazy $b ::= Bool \mid a = a \mid a \leq a \mid \neg b \mid b \wedge b \mid b \vee b$
- Príkazy: $c ::= skip \mid c; c \mid if\ b\ then\ c\ else\ c \mid while\ b\ do\ c$

2.7 Operačná sémantika

- Pomocou odvodzovacieho systému popisuje ako program počíta.
- BigStep: Konfigurácia je valuácia premenných. Prechody sú označené príkazmi a existujú ak sú dokázateľné.
- Tvar odvodzovacieho systému zodpovedá vykonaniu celého príkazy (artimetika sa vyhodnocuje na čísla, Pravdivosť na 1/0, while sa vyhodnotí rekurzívne)
- SmallStep: Konfigurácia je zostávajúci program + valuácia premenných. Prechody nie sú označené a reprezentujú jednotlivé inštrukcie programu.
- Tvar odvodzovacieho systému zodpovedá vykonaniu jednej inštrukcie (idem z ľava do prava a redukujem list AST stromu, while sa vyhodnotí na jeden unroll, vyhodnotený príkaz sa zmení na skip, kompozíciu skipo + niečo viem zameniť za niečo)
- While môže mať nekonečný dôkaz ak cyklus nekončí (prechod neexistuje)

2.8 Denotačná sémantika

- Pomocou funkcie popisujem čo program počíta.
- Funkcia mi pre program vracia funkciu transformácie stavu (res. výpočtu hodnoty na základe stavu pre artimetiku a bool).
- Funkcia je definovaná intuitívne, jediný problém je s while. $\mathcal{C}[while]$ je least fixed point funkcie $\Gamma(f) = \{(a, a) \mid \mathcal{B}(a) = \bot\} \cup \{(a, b) \mid \mathcal{B}(a) = \top \wedge b = (f \circ \mathcal{C}[c])(a)\}$ (funkcia je monotónna a zachováva supréma, teda je Scott-continuous a teda môžem použiť Kleeneho).

2.9 Axiomatická sémantika (Hoareho logika)

- Nehovorí nič o ukončení programu! Len parciálna korektnosť.
- Hoareho trojica $A \mid c \mid B$ — Ak platí A a c skončí, bude platiť B.
- Hoareho odvodzovacý systém:
  - Axióm: $A \mid skip \mid A$
  - Axióm: $A[X/t] \mid X = t \mid A$
  - Rule: $A \mid a; b \mid B$ ak $A \mid a \mid C$ a $C \mid b \mid B$
  - Rule: $A \mid if\ b\ then\ c\ else\ d \mid B$ ak $(A \wedge b) \mid c \mid B$ a $(A \wedge \neg b) \mid d \mid B$
  - Rule: $A \mid while\ b\ do\ c \mid (A \wedge \neg b)$ ak $(A \wedge b) \mid c \mid A$ (A je invariant)

- Rule: $A \mid c \mid B$ ak $\models (A \implies A')$, $\models (B' \implies B)$ a $A' \mid c \mid B'$

- Systém je korektný a úplný (vzhľadom na partial correctness)
- Úplnosť vyplýva z existencie weakest precondition.
- Weakest precondition: Pre statement a postcondition určí najslabšiu vstupnú podmienku. Teda $\vdash WP \mid c \mid B$ (Z WP viem dokázať B po c)
- Weakest precondition vždy existuje! (pri dôkaze while používam Godelov predikát)
- Konštrukcia WP:
  - WP pre $skip, B$ je B
  - WP pre $X = a, B$ je $B[X/a]$
  - WP pre $a; b, B$ je WP pre a z WP pre b, B
  - WP pre if, B je $(b \implies WP(c, B)) \wedge (\neg b \implies WP(d, B))$
  - WP pre $while, B$ je $(b \wedge I) \implies WP(c, I)$ a $(\neg b \wedge I) \implies B$ (pozor, tie dve formule chcem kvantifikovať oddelene)

2.10 LTL model checking

- Kripkeho štruktúra na Buchi automat (všetky stavy sú akceptujúce)
- Vlastnosť na Buchi automat (exponenciálnej veľkosti ak mám smolu)
- Na to aby som ukázal $M \models \varphi$, potrebujem $L(M) \subseteq L(\varphi)$, to ale platí iff $L(M) \cap co - L(\varphi) = \emptyset$.
- Zostrojím teda kompozíciu automatov pre M a pre φ a v nej hľadám akceptujúcy cyklus.
- NestedDFS: Dve (súbežné) DFS — jedno mi určuje postorder akceptujúcich stavov, druhé mi z každého akceptujúceho stavu hľadá cyklus. Celková zložitosť je lineárna.
- Sú triedy vlastností ktoré sa overujú jednodušsie: safety — automat je terminálny (cyklus je selfloop), weak LTL (response vlasnosti) (komponenty sú buď akceptujúce alebo nie, nie sú mixed), stačí mi DFS.

2.11 Symbolický model checking

- Stavy sú bitvektory. Mnoéžiny stavov sú množiny bitvektorov. Teda množina stavov je booleovská funkcia.
- OBDD — ordered binary decision diagram — mám usporiadanie na premenných a zlúčil som všetky duplicitné a redundantné vrcholy.
- Minimalizácia: Zlep listy, odstráň vrcholy ktoré majú identický low/high link, spoj dvojice vrcholov ktoré majú rovnaký low/high link (pre rovnakú premennú)
- Aplikácia funkcie — Shannonova expanzia $F(x, ...) = (x \wedge F_{x \to 1}) \vee (\neg x \wedge F_{x \to 0})$ — Postupujem rekurzívne od vrchu BDD až po listy. Keď prídem do listu, aplikujem funkciu, inak len traverzujem graf bližšie k listom (teda simulujem prácu s BDT pomocou grafu BDD). Výsledky potom rekurzívne zliepam a minimalizujem.
- Model checking: Iniciálne stavy sa kódujú ako BDD. Prechodová funkcia sa kóduje ako BDD (dvojnásobná veľkosť, doslova kódujem reláciu prechodu). Jeden krok sa kóduje ako konjunkcia s prechodovou funkciou a potom zahodenie a prvej polovice premenných.
- Bounded model checking: Kódovanie do SMT formule s unrollingom dĺžky k. Nie všetko sa dá takto dokázať (GFa).

2.12 Prevod LTL na Buchi

- Formula obsahuje len X a U, R a negáciu len na literáloch (k tomu potrebujem R)
- Algoritmus najskôr vyrobí graf automatu, z neho urobí rozšírený automat a ten spojí na obecného Buchiho.
- Výroba grafu automatu:
  - Každý stav si pamätá čo má (Now), čo chce (New) a čo potrebuje ($Next$).
  - Začínam s jedným stavom ktorý má v seve ako new celú formulu.
  - Postupne unrollujem podľa pravidla: Ak mám niečo v new, zamením tento stav za (jeden alebo viac) nových. Ak v new nič nie je, vyrobím nového nasledníka ktorý bude mať v new môj next.
  - Ak vyrobím duplicitný stav, tak ho len mergnem.
  - Unrolling propozície len overí či už náhodou neplatí jej negácia (ak ano, uzol sa maže a končím)
  - Unrolling and/or je buď na dva stavy kde je ľavá/pravá ako new, alebo na jeden kde sú obidve ako new.
  - Unrolling Nextu je že sa podformula pridá do next.
  - Unrolling Untilu je na dva stavy, jeden kde je reach v now a jeden kde je path v now a until v next.
  - Unrolling Release je na dva stavy, jeden kde sú obidve v now a jeden kde je path v now a release v next.
- Výroba rozšíreného automatu: Prechody sú pod propozíciami ktoré platia v stavoch kam vedú. Pre každý until akceptujem množinou stavov kde v now buď je reach, alebo v now nie je celý until. (Teda ak raz vbehnem do miesta kde má platiť until, tak musím prísť až do reachu)

2.13 Process Rewrite Systems

-