Šta je SAT

Milan Popović

SAT (skraćeno od **SAT**isfiability) je problem određivanja da li je Bulova formula zadovoljiva ili ne. Ako je formula zadovoljiva, to znači da varijablama u formuli mogu biti dodeljene vrednosti (**TAČNO** ili **NETAČNO**) tako da cela formula dobije vrednost **TAČNO**.

Pre nego što se budemo detaljnije bavili **SAT** problemom, upoznaćemo se sa osnovnim pojmovima jedne posebne grane matematičke logike poznate kao **Bulova logika**.

Bulove formule

Logika nam pomaže da analiziramo i donosimo zaključke o prostim i složenim događajima.

Poseban slučaj logike je Bulova logika u kojoj događaji mogu imati vrednost **0** ili **1** (**TAČNI** ili **NETAČNI**).

Iako naizgled jednostavna, ovakva definicija nam omogućava da analiziramo i vrlo komplikovane logičke konstrukcije.

Jedan način da to postignemo je da počnemo sa prostim (atomizirnim) događajima, za koje lako možemo utvrditi da li su TAČNI ili NETAČNI. Od takvih atomiziranih događaja možemo da konstruišemo komplikovanije formule pomoću logičkih operacija kao što su **i**, **ili**, **ne**. Tako nastale formule nazivamo Bulovim formulama.

Svaka Bulova formula može imati vrednost TAČNO ili NETAČNO u zavisnosti od vrednosti pojedinačnih atomiziranih događaja koje opisuje i operacija kojima su ti događaji povezani u formuli.

Sintaksa Bulovih formula

Uprošćeno rečeno, Bulova formula je konačan niz slova koji je konstruisan od:

- Varijabli uzetih iz nekog skupa varijabli, nazovimo ga VAR.
  Na primer VAR = { x1, x2, ...xn },
- Bulovih operacija:
  $\neg$ (negacija), $\vee$ (disjunkcija, "ili"), $\wedge$ (konjukcija, "i"), $\rightarrow$ (implikacija), $\leftrightarrow$ (ekvivalencija),
- Zagrada: (,)

Rekurzivna definicija Bulovih formula

Definicija: Skup Bulovih formula (nazovimo ga **BF**) je skup sa sledećim elementima:

- Svaka varijabla je Bulova formula. To jest, VAR $\subseteq$ BF.
- Ako su $\psi 1$, $\psi 2$ u skupu BF , onda su u BF i sledeći izrazi:
  $(\neg\psi 1)$, $(\psi 1 \vee \psi 2)$, $(\psi 1 \wedge \psi 2)$, $(\psi 1 \rightarrow \psi 2)$, i $(\psi 1 \leftrightarrow \psi 2)$.

Primer: $\phi = ((x1 \wedge (\neg x2)) \rightarrow ((\neg x3) \wedge x2))$ je Bulova formula.

Ponekad pišemo $\phi(x1, x2, x3)$ umesto ϕ da naznačimo koje se sve varijable pojavljuju u formuli.

Semantika Bulovih formula

Svakoj varijabli u Bulovoj formuli može biti dodeljena jedna od dve moguće vrednosti: **1** (**TAČNO**) ili **0** (**NETAČNO**).

Formalno gledano, dodeljivanje vrednsoti varijablama je funkcija $\tau : VAR^n \rightarrow \{0, 1\}$.

Podsećamo da postoji 2^n mogućih načina dodele vrednosti za Bulovu formulu sa n varijabli.

Za svaku dodelu τ možemo odrediti vrednost Bulove formule. Ta vrednost će biti TAČNO (1) ili NETAČNO (0). To znači da Bulova formula predstavlja funkciju kojom se skup svih mogućih dodela vrednosti varijablama preslikava u skup {0, 1}, to jest skup $\{\tau : \{0,1\}^n \rightarrow \{0, 1\}\}$.

U skladu sa gornjim definicijama, sledeća procedura rekurzivno izračunava Bulove funkcije:

1. Ako je ϕ varijabla, vredost za ϕ se izračunava tako što pogledamo vrednost koja je dodeljena toj varijabli. Drugim rečina, $\phi(\tau) = \tau(\phi)$.

2. Ako je ϕ složena formula (koja sadrži formule $\psi 1$ i $\psi 2$), tada:

- Ako je $\phi = (\neg\psi 1)$ onda je $\phi(\tau) = 1$ akko (ako i samo ako) je $\psi 1(\tau) = 0$.
- Ako je $\phi = (\psi 1 \vee \psi 2)$ onda je $\phi(\tau) = 1$ akko je $\psi 1(\tau) = 1$ ili $\psi 2(\tau) = 1$.
- Ako je $\phi = (\psi 1 \wedge \psi 2)$ onda je $\phi(\tau) = 1$ akko je $\psi 1(\tau) = 1$ i $\psi 2(\tau) = 1$.
- Ako je $\phi = (\psi 1 \rightarrow \psi 2)$ onda je $\phi(\tau) = 1$ akko je $\psi 1(\tau\ ) = 0$ ili $\psi 2(\tau\ ) = 1$.
- Ako je $\phi = (\psi 1 \leftrightarrow \psi 2)$ onda je $\phi(\tau) = 1$ akko je $(\psi 1(\tau) = 1$ i $\psi 2(\tau\ ) = 1)$ ili $(\psi 1(\tau\ ) = 0$ i $\psi 2(\tau\ ) = 0)$.

Primer: Neka je $\phi = ((x1 \wedge (\neg x2)) \rightarrow ((\neg x3) \wedge x2))$. Neka je $\tau = \{x1 \rightarrow 1, x2 \rightarrow 0, x3 \rightarrow 1\}$, to jest varijablama x1 i x3 smo dodelili vrednost TAĆNO a varijabli x2 vrednost NETAČNO. Lako je pokazati, korišćenjem gornjih pravila da je za datu dodelu $\phi(\tau) = 0$.

U mnogim slučajevima se lako osloboditi zagrada (ukoliko stvarno nisu neophodne). Ako ne koristimo zagrade onda moramo dati prvenstvo operaciji $\neg$ nad drugim Bulovim operacijama. Jasno je da je $((x1 \wedge x2) \wedge x3)$ isto što i $(x1 \wedge (x2 \wedge x3))$ (slično važi i za operaciju $\vee$). Zato umesto višestrukih zagrada možemo da pišemo prosto $(x1 \wedge x2 \wedge x3)$.

Bulove funkcije možemo posmatrati i kao način da kompaktno prikažemo skup svih dodela varijablama za koje Bulova funkcija ima vrednost TAČNO (1).

Primer: Posmatrajmo skup { (0,1), (1,1) } koji predstavlja skup vrednosti varijabli x1, x2 za koje formula F ima vrednost 1. Koja je to formula?

Lako je pokazati da je $F = (\neg x1 \wedge x2) \vee (x1 \wedge x2)$.

Čest je slučaj da dve formule koje koriste različite Bulove operacija predstavljaju isti skup dodela vredosti. To nas upućuje na sledeću definiciju:

Definicija: Dve Bulove formule su ekvivalentne ako imaju istu vrednost za sve moguće dodele vrednosti varijablama. Drugim rečima kažemo da su ϕ i ψ ekvivalentne (i pišemo $\phi \equiv \psi$) ako za svako $\tau : VAR^n \rightarrow \{0, 1\}$ imamo da je $\phi(\tau) = \psi(\tau)$.

Primer: $\phi 1 = ((x1 \rightarrow x2) \wedge x1)$ je ekvivalentno sa $\phi 2 = (x1 \wedge x2)$.

Konjuktivna normalna forma

Konjuktivna normalna forma (**KNF**) je specijalan oblik Bulovih formula koji se smatra najpogodnijim oblikom pri rešavanju SAT problema. Sve Bulove formule se, kao što ćemo videti, mogu prikazati u KNF obliku.

Definicija: Literal je varijabla (pozitivan literal) ili njena negacija (negativan literal)

Primer: p, $\neg$q, x1, $\neg$x8 su literali.

Definicija: Klauzula je:

- Literal, ili
- Disjunkcija jednog ili više literala, ili
- Prazna klauzula, koju ćemo označiti sa ().

Primer: p, $p \vee r$, $(p \vee \neg q \vee r)$, $(x1 \vee x2 \vee x3 \vee \neg x4)$, () su klauzule.

Definicija: Bulova formula ϕ je u konjuktivno normalnoj formi (KNF) ako je konjukcija jedne ili više klauzula.

Primer: $(p \vee q) \wedge (q \vee \neg s \vee r) \wedge (\neg r \vee t)$ je u KNF, ima tri klauzule povezane konjukcijom.

Konverzija u KNF

Sldeća procedura konvertuje Bulovu formulu u KNF:

1. U formuli zamenimo sve implikacije i ekvivalencije:

- Koristimo $(p \rightarrow q) \equiv (\neg p \vee q)$ da implikaciju zamenimo klauzulom.
- Koristimo $(p \leftrightarrow q) \equiv (p \rightarrow q) \wedge (q \rightarrow p) \equiv (\neg p \vee q) \wedge (\neg q \vee p)$ da ekvivalenciju zamenimo konjukcijom dve klauzule.

2. Koristimo De Morganova pravila da zamenimo negacije:

- $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$
- $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$

3. Koristimo distribuciju disjunkcije u odnosu na konjukciju: $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$

Primer: Konvertovati $p \leftrightarrow (r \wedge s)$ u KNF:

- $p \leftrightarrow (r \wedge s) => (p \vee \neg(r \wedge s)) \wedge (\neg p \vee (r \wedge s))$ - zamena ekvivalencije.
- $(p \vee \neg(r \wedge s)) \wedge (\neg p \vee (r \wedge s)) => (p \vee \neg r \vee \neg s) \wedge (\neg p \vee (r \wedge s))$ - zamena negacije $\neg(r \wedge s)$ sa $(\neg r \vee \neg s)$
- $(p \vee \neg r \vee \neg)) \wedge (\neg p \vee (r \wedge s)) => (p \vee \neg r \vee \neg s) \wedge (\neg p \vee r) \wedge (\neg p \vee s)$ - distribucija $\neg p \vee (r \wedge s)$
- $(p \vee \neg r \vee \neg s) \wedge (\neg p \vee r) \wedge (\neg p \vee s)$ je KNF.

Standardni algoritam za konverziju Bulove formule u KNF zahteva eksponencijalno vreme, zato što se, u najgorem slučaju, Bulova formula konvertuje u 2^n klauzula. Međutim moguće je, u polinomijalnom vremenu, konverovati bilo koju Bulovu formulu u KNF formulu koja nije striktno ekvivalentna originalnoj formuli ali je zadovoljiva samo ako je originalna formula zadovoljiva.

Zadovoljivost Bulovih formula

U logici i kompjuterskim naukama, problem zadovoljenja Bulove formule je problem kojim se određuje da li neka Bulova formula može biti zadovoljena u nekoj interpretaciji.
Drugim rečima, postavlja se pitanje da li se varijablama koje se pojavljuju u Bulovoj formuli mogu konzistentno dodeliti vrednosti TAČNO ili NETAČNO tako da cela formula dobije vrednost TAČNO.

- Ako takva dodela (interpretacija) postoji za Bulovu formulu se kaže da je zadovoljiva (satisfiable).
- Ako takva zamena ne postoji, to jest formula ima vrednost NETAČNO za sve moguće dodele (interpretacije), onda za takvu formulu kažemo da je nezadovoljiva (unsatisfiable).

Gornji problem je poznat kao SAT problem i predstavlja jedan od osnovnih problema u teoriji složenosti algoritama.

SAT je bio i prvi problem za kojeg je dokazano da je NP-kompletan (Cook-Levin teorema). To znači da su svi problemi iz NP klase manje ili jednako teški kao SAT. Pitanje da li se SAT može rešiti algoritmom polinomijalne vremenske složenosti je ekvivalentno pitanju da li je P = NP. P vs NP je jedan otvoren problem u matematičkoj teoriji izračunljivosti. SAT takođe ima i veoma značajnu primenu u mnogim praktičnim aplikacijama. Sve to SAT problemu daje izuzetno veliki značaj.

Do sada je razvijeno više heurističkih SAT algoritama koji su u stanju da reše SAT probleme sa desetinama hiljada varijabli i formula sastavljenih od miliona operacija, što je dovoljno za rešavanje mnogih praktičnih zadataka iz oblasti kao što su veštačka inteligencija, dizajn digitalnih kola i automatsko dokazivanje teorema.

Mi ćemo se nadalje baviti algoritmima za rešavanje SAT problema. Svi poznati algoritni se uglavnom baziraju na osnovnom DPLL algoritmu (Davis-Putnam–Logemann–Loveland). Zato ćemo i početi sa razmatranjem ovog algoritma kao i sa jednom njegovom implementacijom u Python jeziku.

SAT algoritmi

Pre nego što pređemo na razmatranje drugih algoritama prikazaćemo jednostavan "brute-force" algoritam kojim se SAT problem rešava ispitivanjem svih mogućih dodela vrednosti varijablama Bulove formule.

Brute-force sat algoritam

Neka je formula(x1, x2, x3, ..., xn) bulova formula od n varijabli. Kako je ranije navedeno broj različitih dodela vrednosti {True, False} varijablama x1, x2, x3, ... , xn je 2^n. Zato "brute-force" algoritam ima eksponencijalnu vremensku kompleksnost $O(2^n)$.

Python-like "brute-force" sat algoritam.

```
def brute_force_sat(formula, n):
    for varijable in itertools.product([True,False], repeat=n):
        if formula(varijable):
            return True
    return False
```

Gornji algoritam vraća True ako je formula zadovoljiva, vraća False ako nije. U najgorem slučaju algoritmu je potrebno 2^n koraka da obavi celu for petlju iz linije 2. Za algoritme koji

zahtevaju eksponencijalan broj koraka kažemo da su NP (non-polynomial, ne-polinomski) i smatramo ih praktično neupotrebljivim (intractible) u slučaju velikog broja varijabli (velikog inputa). Recimo da gornji algoritam za 100 varijabli zahteva 2^{100} koraka, što je ogroman broj (1267650600228229401496703205376 sa 31 cifrom). To ne mogu da postignu ni najbrži računari.

Kao što je već rečeno, u skorije vreme su razvijeni heuristički algoritmi koji rešavaju Bulove formule (SAT probleme) sa više desetina hiljada varijabli. Pa, kako je to moguće?

Davis-Putnam (DP) procedura

Kao ulaz u DP algoritam koristi se Bulova formula u KNF, odnosno liste klauzula.

Sledeća pravila se se koriste sve dok se mogu primeniti na listi klauzula:

1. **Pravilo jednog literala:** Ukloniti sve klauzule sa jednim literalom, kao i sve druge klauzule koje sadrže taj literal. Ukloniti negaciju tog literala iz svih klauzula. Ovo pravilo je poznato i kao jedinična propagacija.
2. **Pravilo čistih literala:** Odrediti skup svih litarala koji se u klauzulama pojavljuju kao samo pozitivni ili samo negativni. Takve literale nazivamo čistim. Ukloniti sve klauzule koje imaju čiste literale.
3. **Pravilo rezolucije:**

    1. Odaberemo literal koji se u listi klauzula pojavljuje i kao pozitivan (l) i kao negativan ($\neg l$) .
    2. Kreiramo listu C koja sadrži klauzule u kojima se pojavljuje l i uklonimo l iz tih klauzula.
    3. Kreiramo listu D koja sadrži klauzule u kojima se pojavljej $\neg l$ i uklonimo $\neg l$ iz tih klauzula.
    4. kombinijemo liste C i D tako što svaku klauzulu iz C proširimo svakom klauzulom iz D i uklonimo duple klauzule kao i klauzule koje sadrže $l \vee \neg l$.
    5. Kreiramo listu X koja sadrži klauzule u kojima se ne pojavljuju ni l ni $\neg l$.

4. **Ponovimo korake 1, 2 i 3** sa novom listom X + C + D klauzula: Procedura se zaustavlja kada više nema klauzula, što označava da je fomula zadovoljiva, ili kad se u listi klauzula pojavi prazna klauzula, što označava da je formula nezadovoljiva.

Primer 1. Proveriti da li je $p \wedge (p \vee q) \wedge (\neg p \vee \neg q) \wedge (q \vee r) \wedge (\neg q \vee \neg r)$ zadovoljiva:

> Korak 1. Primenom pravila 1. za literal p, formula se svodi na: $\neg q \wedge (q \vee r) \wedge (\neg q \vee \neg r)$.
> Korak 2. Ponovo primenjujemo pravilo 1. za literal $\neg q$, pa formula postaje r.
> Korak 3. Ponovo primenjujemo pravilo 1. za r. Formula nema više klauzula, dakle zadovoljiva je.

Primer 2. Proveriti da li je formula (p $\vee$ r) $\wedge$ (p $\vee$ $\neg$s) $\wedge$ ($\neg$p V s) $\wedge$ ($\neg$p V $\neg$r) $\wedge$ (s V $\neg$r) $\wedge$ ($\neg$s V r) zadovoljiva.

Korak 1. Primenjujemo pravilo 3.1 i odabiramo literal p.
Korak 2. Primenimo pravilo 3.2 za literal p. Dobijamo listu C : r, $\neg$s
Korak 3. Primenimo pravilo 3.3 za literal $\neg$p. Dobijamo listu D: s, $\neg$r
Korak 4. Primenimo pravilo 3.4. Dobijamo formulu :
(r $\vee$ s) $\wedge$ (r $\vee$ $\neg$r) $\wedge$ ($\neg$s $\vee$ s) $\wedge$ ($\neg$s $\vee$ $\neg$r) => (r $\vee$ s) $\wedge$ ($\neg$s $\vee$ $\neg$r)
Korak 5. Primenimo pravilo 3.5 da kreiramo:
(s V $\neg$r) $\wedge$ ($\neg$s V r).
Korak 6. Primenimo pravilo 4. sa novom listom:
(r $\vee$ s) $\wedge$ ($\neg$s $\vee$ $\neg$r) $\wedge$ (s V $\neg$r) $\wedge$ ($\neg$s V r)
Korak 7. Primenimo pravilo 3.1 i odabiramo literal r.
Korak 7. Primenimo pravilo 3.2 za literal r. Kreiramo listu C : s, $\neg$s
Korak 8. Primenimo pravilo 3.3 za literal $\neg$r. Kreiramo listu: $\neg$s, s
Korak 9. Primenimo pravilo 3.4. Dobijamo:
(s $\vee$ $\neg$s) $\wedge$ (s $\vee$ s) $\vee$ ($\neg$s $\vee$ $\neg$s) $\wedge$ ($\neg$s $\vee$ s) => s $\wedge$ $\neg$s što je nemoguće zadovoljiti.

Davis-Putnam-Longemann-Loveland (DPLL) procedura

DPLL procedura je vrlo jednostavna. Zasniva se na dodjeljivanju vrednosti varijablama jednu po jednu, i pojednostavljivanju formule u svakom koraku.
Na primer, ako formula sadrži klauzulu x3 $\vee$ $\neg$x5 i upravo smo dodelili varijabli x5 T (tj. TAČNO), tada klauzula postaje istina i može se ukloniti. Obrnuto, ako smo varijabli x5 dodelili F (NETAČNO) onda je jedini način kako preostale klauzule mogu biti zadovoljene je ako je x3 = T. Dakle, x5 = F iziskuje x3 = T. Ovakve zamene pojednostavljuju formule.

Recimo da je zadata folrmula ϕ. Odaberimo varijablu, recimo xi. Zamenimo xi = T u ϕ i pojednostavimo ϕ. Rekurzivno proverimo da li je pojednostavljena formula zadovoljiva. Ako se ispostavi da je nezadovoljiva, zamenimo xi = F u ϕ, ponovo pojednostavimo ϕ i rekurzivno proverimo zadovoljivost preostalih klauzula. Ako i to ispadne nezadovoljivo, onda proglasite ϕ nezadovoljivom. Ako uspešno izvršimo zamenu za sve varijable, formula je zadovoljiva.

Pri implementaciji DPLL algoritma postoje različiti izbori. Na primjer, kojim redom birati varijable? Slučajno, ili onu koja se pojavljuje u većini klauzula, itd. Slično, da li prvo isprobati vriednost T ili F? Koju strukturu podataka koristiti za praćenje varijabli i klauzula? Mnoge varijante su proučavane i iznenađujuće, vrlo dobro su se pokazale u praksi.

Najuspješnije varijante ovog algoritma uključuju mašinsko učenje. Pretpostavimo da formula ima klauzule (x1 $\vee$ x7 $\vee$ x9) i (x1 $\vee$ $\neg$x9 $\vee$ $\neg$x6) i u nekoj grani algoritam je pokušao x1 = F, x7 = F, x6 = T, što je dovelo do kontradikcije jer x9 se prisiljava da bude i T i F. Time je

algoritam saznao da je ova kombinacija zabranjena, ne samo u ovom trenutku već i na svim drugim granama koje će ispitivati u budućnosti. Ovo znanje se može dodati u obliku nove klauzule x1 ∨ x7 ∨ ¬x6, budući da svaka dodela varijabli koja zadovoljava početnu formulu mora zadovoljiti i ovu novododatu klauzulu. Kao što se može zamisliti, učenje klauzula dolazi u bezbroj varijanti, zavisno od toga koje se pravilo koristi za zaključivanje i dodavanje novih klauzula.

Možemo se zapitati kako dodavanje klauzula (tj. više ograničenja) pojednostavljuje problem umesto da ga usložnjava. Odgovor je da se klauzule mogu smatrati smernicama ka zadovoljenju formule.

DPLL algoritam

1. **Pravilo jednog literala:** Ukloniti sve klauzule sa jednim literalom, kao i sve druge klauzule koje sadrže taj literal. Ukloniti negaciju tog literala iz svih klauzula. Ovo pravilo je poznato i kao jedinična propagacija.

2. **Pravilo čistih literala:** Odrediti skup svih litarala koji se u klauzulama pojavljuju kao samo pozitivni ili samo negativni. Takve literale nazivamo čistim. Ukloniti sve klauzule koje imaju čiste literale.

3. **Pravilo razdvajanja:** Izaberimo jedan literal l iz formule ϕ. Formiramo dve formule, jednu tako što originalnoj formuli dodamo l $(l \wedge \phi)$ i drugu tako što formuli dodamo l $((\neg l \wedge \phi)$. Sada rekurzivno primenimo algoritam na obe nove formule. Procedura se zaustavlja kada više nema klauzula, što označava da je fomula zadovoljiva, ili kad se u listi klauzula pojavi prazna klauzula, što označava da je formula nezadovoljiva.

Primer 1. Proveriti da li je $p \wedge (p \vee q) \wedge (\neg p \vee \neg q) \wedge (q \vee r) \wedge (\neg q \vee \neg r)$ zadovoljiva:

- Korak 1. Primenimo Pravilo 1. za literal p. U formuli $p \wedge (p \vee q) \wedge (\neg p \vee \neg q) \wedge (q \vee r) \wedge (\neg q \vee \neg r)$ stavimo $p = T$.
  Formula se pojednostavljuje i postaje: $\neg q \wedge (q \vee r) \wedge (\neg q \vee \neg r)$
- Korak 2. Primenimo pravilo 1. za literal ¬q. U formuli $\neg q \wedge (q \vee r) \wedge (\neg q \vee \neg r)$ stavimo $q = T$.
  Formula postaje $\wedge\, r \wedge \neg r \equiv$ False. Ne može se zadovoljiti. Moramo pokušati sa $q = F$.
- Korak 3. Primenimo pravilo 1. za literal $q = F$ u formuli $\neg q \wedge (q \vee r) \wedge (\neg q \vee \neg r)$. Formula postaje $T \wedge r \equiv r$.
- Korak 4. Primenimo pravilo 1. U formuli r odaberimo $r = T$.
  Formula postaje T. Dakle formula je zadovoljiva.

Da smo u koraku 1.1 odabrali $q = F$ (umesto $q = T$) algoritam bi završio sa jednim korakom manje. To pokazuje zašto je značajan redosled izbora vrednosti varijable.

Primer 2. Proveriti da li je formula $(p \vee r) \wedge (p \vee \neg s) \wedge (\neg p \text{ V } s) \wedge (\neg p \text{ V } \neg r) \wedge (s \text{ V } \neg r) \wedge (\neg s \text{ V } r)$ zadovoljiva.

Literatura:

1. Handbook of Satisfiability. Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh (2009)
2. DP: A Computing Procedure for Quantification Theory. Martin Davis, Hilary Putnam (1960).
3. DPLL: A Machine Program for Theorem Proving. Martin Davis, George Logemann, Donald Loveland (1962).
4. SAT Solvers: Theory and Practice. Clark Barrett
5. SAT-Solving: From Davis-Putnam to Zchaff and Beyond (SAT Basics). Lintao Zhang