7 AGEN LOGIS

Di mana kami merancang agen yang dapat membentuk representasi dunia yang kompleks, gunakan a proses inferensi untuk memperoleh representasi baru tentang dunia, dan gunakan ini representasi baru untuk menyimpulkan apa yang harus dilakukan.

Manusia, tampaknya, tahu banyak hal; dan apa yang mereka ketahui membantu mereka melakukan banyak hal. Ini adalah bukan pernyataan kosong. Mereka membuat klaim kuat tentang bagaimana kecerdasan manusia tercapai — bukan dengan mekanisme refleks murni tetapi melalui proses **penalaran** yang beroperasi **representasi** pengetahuan internal . Dalam AI, pendekatan kecerdasan ini diwujudkan dalam **agen berbasis pengetahuan** .

PEMIKIRAN

PERWAKILAN

BERBASIS PENGETAHUAN AGEN

Agen pemecahan masalah Bab 3 dan 4 tahu hal-hal, tetapi hanya dalam sangat terbatas, rasa tidak fleksibel. Sebagai contoh, model transisi untuk 8-puzzle — pengetahuan tentang apa itu tindakan — disembunyikan di dalam kode khusus domain dari fungsi R ESULT . Itu bisa saja digunakan untuk memprediksi hasil tindakan tetapi tidak menyimpulkan bahwa dua ubin tidak dapat menempati ruang yang sama atau yang menyatakan dengan paritas ganjil tidak dapat dicapai dari keadaan dengan paritas genap. Itu representasi atom yang digunakan oleh agen pemecahan masalah juga sangat terbatas. Secara parsial lingkungan yang dapat diamati, satu-satunya pilihan agen untuk mewakili apa yang diketahui tentang kondisi saat ini adalah untuk membuat daftar semua keadaan konkret yang mungkin — prospek tanpa harapan di lingkungan besar.

Bab 6 memperkenalkan gagasan mewakili negara sebagai penugasan nilai untuk berbagai mampu; ini adalah langkah ke arah yang benar, memungkinkan beberapa bagian agen untuk bekerja di a cara independen domain dan memungkinkan algoritma yang lebih efisien. Dalam bab ini dan yang mengikuti, kita mengambil langkah ini sampai pada kesimpulan logisnya, begitulah - kita mengembangkan **logika** sebagai kelas umum representasi untuk mendukung agen berbasis pengetahuan. Agen semacam itu bisa menggabungkan dan menggabungkan kembali informasi agar sesuai dengan berbagai tujuan. Seringkali, proses ini bisa sangat jauh dari kebutuhan saat — seperti ketika seorang ahli matematika membuktikan teorema atau seorang astronom menghitung harapan hidup bumi. Agen berbasis pengetahuan dapat menerima yang baru tugas dalam bentuk tujuan yang dijelaskan secara eksplisit; mereka dapat mencapai kompetensi dengan cepat dengan menjadi diceritakan atau belajar pengetahuan baru tentang lingkungan; dan mereka dapat beradaptasi dengan perubahan dalam lingkungan dengan memperbarui pengetahuan yang relevan.

LOGIKA

Kita mulai di Bagian 7.1 dengan desain agen keseluruhan. Bagian 7.2 memperkenalkan sim- di lingkungan baru, dunia wumpus, dan mengilustrasikan operasi berbasis pengetahuan agen tanpa masuk ke detail teknis. Kemudian kami menjelaskan prinsip-prinsip umum **logika**

dalam Bagian 7.3 dan spesifik **logika proposisional** dalam Bagian 7.4. Meski kurang ekspresif dari **logika tingkat pertama** (Bab 8), logika proposisional menggambarkan semua konsep dasar logika; itu juga dilengkapi dengan teknologi inferensi yang dikembangkan dengan baik, yang kami jelaskan dalam bagian tions 7.5 dan 7.6. Akhirnya, Bagian 7.7 menggabungkan konsep agen berbasis pengetahuan dengan teknologi logika proposisional untuk membangun beberapa agen sederhana untuk dunia wumpus.

7.1 K NOWLEDGE - B ASED A GENTS

DASAR PENGETAHUAN
KALIMAT
PENGETAHUAN PERWAKILAN BAHASA AKSIOMA

Komponen utama agen berbasis **pengetahuan adalah basis pengetahuannya** , atau KB. Sebuah pengetahuan base edge adalah seperangkat **kalimat** . (Di sini "kalimat" digunakan sebagai istilah teknis. Itu terkait tetapi tidak identik dengan kalimat bahasa Inggris dan bahasa alami lainnya.) Setiap kalimat diekspresikan dalam bahasa yang disebut **bahasa representasi pengetahuan** dan mewakili beberapa pernyataan tentang dunia. Terkadang kita menghargai kalimat dengan nama **aksioma** , ketika kalimat diambil seperti yang diberikan tanpa diturunkan dari kalimat lain.

Harus ada cara untuk menambahkan kalimat baru ke basis pengetahuan dan cara untuk bertanya apa yang diketahui. Nama standar untuk operasi ini adalah T ELL dan A SK , masing-masing.

KESIMPULAN

Kedua operasi itu mungkin melibatkan **inferensi** — yaitu, mendapatkan kalimat baru dari yang lama. Kesimpulan harus mematuhi persyaratan bahwa ketika salah satu A SK s pertanyaan dari basis pengetahuan, jawabannya harus mengikuti dari apa yang telah dikatakan (atau T ELL ed) ke basis pengetahuan sebelumnya. Kemudian dalam bab ini, kita akan lebih tepat tentang kata penting "ikuti." Untuk saat ini, bawa ke berarti bahwa proses inferensi tidak harus memperbaiki keadaan seiring berjalannya waktu.

Gambar 7.1 menunjukkan garis besar program agen berbasis pengetahuan. Seperti semua agen kami, dibutuhkan persepsi sebagai input dan mengembalikan suatu tindakan. Agen mempertahankan basis pengetahuan, KB,

LATAR BELAKANG PENGETAHUAN

yang awalnya mungkin berisi beberapa **latar belakang pengetahuan** .

Setiap kali program agen dipanggil, ia melakukan tiga hal. Pertama, T ELL s Knowledge yang dasar tepi apa yang dirasakannya. Kedua, SK adalah basis pengetahuan tindakan apa yang harus dilakukan melakukan. Dalam proses menjawab pertanyaan ini, penalaran yang luas dapat dilakukan keadaan dunia saat ini, tentang hasil dari kemungkinan urutan tindakan, dan sebagainya. Ketiga, program agen T ELL adalah basis pengetahuan tindakan yang dipilih, dan agen mengeksekusi tindakan.

Rincian bahasa representasi disembunyikan di dalam tiga fungsi yang mengimplementasikan ment antarmuka antara sensor dan aktuator di satu sisi dan representasi inti dan sistem penalaran di sisi lain. M AKE -P ERCEPT -S ENTENCE menyusun kalimat sebagai- menyatakan bahwa agen mempersepsikan persepsi yang diberikan pada waktu tertentu. M AKE -A CTION -Q UERY menyusun kalimat yang menanyakan tindakan apa yang harus dilakukan pada saat ini. Akhirnya, M AKE -A CTION -S ENTENCE menyusun kalimat yang menyatakan bahwa tindakan yang dipilih adalah diamputasi. Rincian mekanisme inferensi disembunyikan di dalam T ELL dan A SK . Kemudian bagian akan mengungkapkan detail ini.

Agen dalam Gambar 7.1 tampak sangat mirip dengan agen dengan kondisi internal yang dijelaskan dalam Bab 2. Karena definisi T ELL dan SK , bagaimanapun, berbasis pengetahuan agen bukanlah program sewenang-wenang untuk menghitung tindakan. Dapat menerima uraian di

Halaman 3


```
fungsi KB-A GENT (percept) mengembalikan suatu tindakan
  gigih : KB, basis pengetahuan
          t, sebuah penghitung, awalnya 0, menunjukkan waktu

  T ELL (KB, M AKE -P ERCEPT -S ENTENCE (persepsi, t))
  aksi ← A SK (KB, M AKE -A CTION -Q UERY (t))
  T ELL (KB, M AKE -A CTION -S ENTENCE (aksi, t))
  t ← t + 1
```

kembali tindakan

Gambar 7.1 Agen berbasis pengetahuan umum. Diberikan percept, agen menambahkan percept ke basis pengetahuannya, meminta basis pengetahuan untuk tindakan terbaik, dan memberi tahu pengetahuan itu mendasarkan bahwa sebenarnya telah mengambil tindakan itu.

TINGKAT PENGETAHUAN

yang **tingkat pengetahuan** , di mana kita perlu menentukan hanya apa agen tahu dan apa tujuannya adalah, untuk memperbaiki perilakunya. Misalnya, taksi otomatis mungkin memiliki tujuan mengambil penumpang dari San Francisco ke Marin County dan mungkin tahu bahwa Golden Jembatan Gerbang adalah satu-satunya penghubung antara kedua lokasi. Maka kita bisa berharap untuk menyeberang Jembatan Golden Gate *karena tahu bahwa itu akan mencapai tujuannya* . Perhatikan bahwa analisis ini

PENERAPAN TINGKAT

tidak tergantung pada bagaimana taksi bekerja di **tingkat implementasi** . Tidak masalah apakah pengetahuan geografisnya diimplementasikan sebagai daftar tertaut atau peta piksel, atau apakah alasannya dengan memanipulasi string simbol yang disimpan dalam register atau dengan menyebarkan sinyal bising dalam a jaringan neuron.

Seorang agen berbasis pengetahuan dapat dibangun hanya dengan T ELL ing itu apa yang dibutuhkan untuk tahu. Dimulai dengan basis pengetahuan yang kosong, perancang agen dapat membuat kalimat ELL satu per satu

DEKLARATIF

sampai agen tahu bagaimana beroperasi di lingkungannya. Ini disebut aplikasi **deklaratif** mendekati pembangunan sistem. Sebaliknya, pendekatan **prosedural** mengkodekan perilaku yang diinginkan langsung sebagai kode program. Pada 1970-an dan 1980-an, para pendukung kedua pendekatan terlibat dalam perdebatan sengit. Kami sekarang memahami bahwa agen yang sukses sering kali menggabungkan kedua deklaratif dan elemen prosedural dalam desainnya, dan bahwa pengetahuan deklaratif sering dapat dikompilasi menjadi kode prosedural yang lebih efisien.

Kami juga dapat menyediakan agen berbasis pengetahuan dengan mekanisme yang memungkinkannya untuk belajar untuk dirinya sendiri. Mekanisme ini, yang dibahas dalam Bab 18, menciptakan pengetahuan umum tentang lingkungan dari serangkaian persepsi. Agen pembelajaran dapat sepenuhnya mandiri.

7.2 T HE W UMPUS W ORLD

DUNIA WUMPUS

Pada bagian ini kami menggambarkan suatu lingkungan di mana agen berbasis pengetahuan dapat menunjukkan mereka bernilai. Dunia **wumpus** adalah sebuah gua yang terdiri dari kamar-kamar yang dihubungkan oleh lorong-lorong. Mengintai di suatu tempat di dalam gua ada wumpus yang mengerikan, binatang buas yang memakan siapa saja yang memasuki kamarnya. Wumpus dapat ditembak oleh agen, tetapi agen hanya memiliki satu panah. Beberapa kamar berisi

Halaman 4

lubang tak berdasar yang akan menjebak siapa pun yang mengembara ke kamar-kamar ini (kecuali untuk wumpus, yang terlalu besar untuk jatuh). Satu-satunya fitur mitigasi dari lingkungan yang suram ini adalah kemungkinan menemukan tumpukan emas. Meskipun dunia wumpus agak jinak oleh modern standar permainan komputer, itu menggambarkan beberapa poin penting tentang kecerdasan.

Contoh dunia wumpus ditunjukkan pada Gambar 7.2. Definisi tugas yang tepat lingkungan diberikan, seperti yang disarankan dalam Bagian 2.3, oleh deskripsi PEAS:

- **Ukuran kinerja** : +1000 untuk memanjat keluar dari gua dengan emas, –1000 untuk jatuh ke dalam lubang atau dimakan oleh wumpus, –1 untuk setiap tindakan yang diambil dan –10 untuk menggunakan panah. Permainan berakhir saat agen mati atau saat agen naik keluar dari gua.
- **Lingkungan** : Kisi-kisi kamar 4 × 4. Agen selalu dimulai di kotak berlabel [1,1], menghadap ke kanan. Lokasi emas dan wumpus dipilih secara acak. domly, dengan distribusi yang seragam, dari kotak selain kotak awal. Di Selain itu, setiap kotak selain awal dapat menjadi lubang, dengan probabilitas 0,2.
- **Aktuator** : Agen dapat bergerak *Maju* , *TurnLeft* hingga 90 $^\circ$, atau *TurnRight* hingga 90°Itu agen meninggal karena sengsara jika memasuki kotak yang berisi lubang atau wumpus hidup. (Saya t aman, meskipun bau, memasuki kotak dengan wumpus mati.) Jika agen mencoba bergerak maju dan menabrak dinding, maka agen tidak bergerak. Tindakan *Grab* bisa digunakan untuk mengambil emas jika berada di kotak yang sama dengan agen. Aksi *Tembak* bisa

digunakan untuk menembakkan panah dalam garis lurus ke arah agen yang dihadapinya. Panah terus sampai itu mengenai (dan karenanya membunuh) wumpus atau mengenai dinding. Agen itu hanya satu panah, jadi hanya aksi *Tembak* pertama yang berpengaruh. Akhirnya, aksi *Climb* dapat digunakan untuk memanjat keluar dari gua, tetapi hanya dari bujur sangkar [1,1].

• **Sensor** : Agen memiliki lima sensor, masing-masing memberikan sedikit informasi:

- Di kotak yang berisi wumpus dan di sebelahnya (tidak diagonal) yang berdekatan kotak, agen akan merasakan *bau busuk* .
- Dalam kotak yang berbatasan langsung dengan lubang, agen akan merasakan *angin* .
- Di kotak di mana emas berada, agen akan melihat *Glitter* .
- Ketika agen berjalan ke dinding, ia akan merasakan *benjolan* .
- Ketika wumpus terbunuh, ia mengeluarkan *Jeritan* menyedihkan yang bisa dirasakan apa saja- di mana di dalam gua.

Persept akan diberikan kepada program agen dalam bentuk daftar lima simbol; misalnya, jika ada bau busuk dan angin sepoi-sepoi, tetapi tidak ada kilau, benjolan, atau teriakan, agen program akan mendapatkan [Stench, Breeze, None, None, None].

Kita dapat menandai lingkungan wumpus di sepanjang berbagai dimensi yang diberikan dalam Bab ter 2. Jelas, itu adalah diskrit, statis, dan agen tunggal. (Wumpus tidak bergerak, untungnya.) Ini berurutan, karena hadiah mungkin datang hanya setelah banyak tindakan diambil. Itu sebagian diamati, karena beberapa aspek negara tidak secara langsung dapat dipahami: lokasi agen kation, kondisi kesehatan wumpus, dan ketersediaan panah. Adapun lokasi lubang dan wumpus: kita bisa memperlakukan mereka sebagai bagian tidak teramati dari negara yang terjadi agar tidak berubah — dalam hal ini, model transisi untuk lingkungan sepenuhnya

Halaman 5

Gambar 7.2 Dunia wumpus yang khas. Agen ada di sudut kiri bawah, menghadap ke kanan.

dikenal; atau kita dapat mengatakan bahwa model transisi itu sendiri tidak diketahui karena agennya tidak tahu tindakan *Maju* mana yang fatal — dalam hal ini, menemukan lokasi lubang dan wumpus melengkapi pengetahuan agen tentang model transisi.

Untuk seorang agen di lingkungan, tantangan utama adalah ketidaktahuan awal tentang figurasi lingkungan; Mengatasi ketidaktahuan ini tampaknya membutuhkan alasan logis. Dalam sebagian besar contoh dunia wumpus, agen mungkin untuk mengambil emas dengan aman. Kadang-kadang, agen harus memilih antara pulang dengan tangan kosong dan mempertaruhkan kematian temukan emasnya. Sekitar 21% dari lingkungan sama sekali tidak adil, karena emas ada di dalam lubang atau dikelilingi oleh lubang.

Mari kita saksikan agen wumpus berbasis pengetahuan yang mengeksplorasi lingkungan yang ditunjukkan pada Gambar 7.2. Kami menggunakan bahasa representasi pengetahuan informal yang terdiri dari tulisan simbol bawah dalam kisi (seperti pada Gambar 7.3 dan 7.4).

Basis pengetahuan awal agen berisi aturan-aturan lingkungan, seperti dijelaskan sebelumnya; khususnya, ia tahu bahwa ia berada di [1,1] dan [1,1] adalah kotak yang aman; kami tunjukkan bahwa dengan "A" dan "OK," masing-masing, di kotak [1,1].

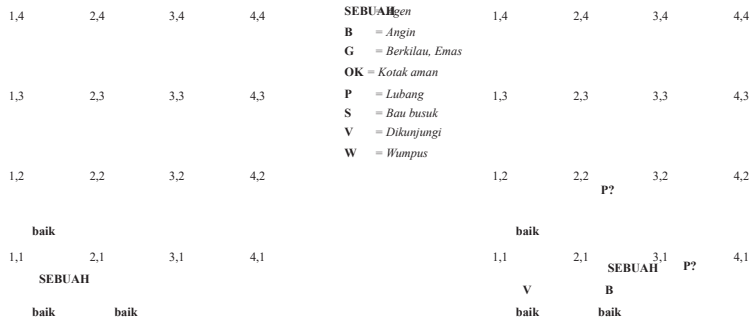
Persepsi pertama adalah [Tidak Ada, Tidak Ada, Tidak Ada, Tidak Ada, Tidak Ada], dari mana agen dapat

menyatakan bahwa bujur sangkar sebelahnya, [1,2] dan [2,1], bebas dari bahaya — semuanya baik-baik saja. Ara-ure 7.3 (a) menunjukkan kondisi pengetahuan agen pada titik ini.

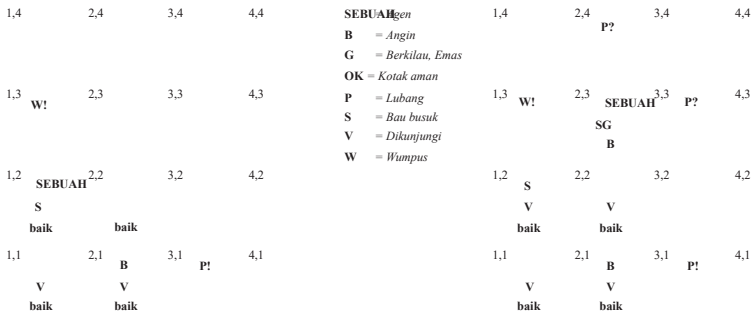
Agen yang berhati-hati hanya akan bergerak ke kotak yang diketahui OK. Mari kita anggap saja agen memutuskan untuk maju ke [2,1]. Agen merasakan angin (dilambangkan dengan "B") di [2,1], jadi harus ada lubang di alun-alun sebelah. Lubang tidak bisa di [1,1], berdasarkan aturan permainan, jadi harus ada lubang di [2,2] atau [3,1] atau keduanya. Notasi "P?" Pada Gambar 7.3 (b) menunjukkan kemungkinan lubang di kotak itu. Pada titik ini, hanya ada satu kotak yang diketahui OK dan itu belum dikunjungi. Jadi agen yang bijaksana akan berbalik, kembali ke [1,1], dan kemudian lanjutkan ke [1,2].

Agen merasakan bau pada [1,2], menghasilkan tingkat pengetahuan yang ditunjukkan pada Gambar 7.4 (a). Bau busuk di [1,2] berarti harus ada wumpus di dekatnya. Tetapi

Halaman 6



Gambar 7.3 Langkah pertama yang diambil oleh agen di dunia wumpus. (a) Situasi awal uation, setelah persept [Tidak, Tidak, Tidak, Tidak, Tidak]. (B) Setelah satu gerakan, dengan persept [Tidak Ada, Breeze, Tidak Ada, Tidak Ada, Tidak Ada].



Gambar 7.4 Dua tahap selanjutnya dalam kemajuan agen. (a) Setelah langkah ketiga, dengan persept [Stench, None, None, None, None]. (B) Setelah langkah kelima, dengan persept [Bau busuk, Angin, Berkilauan, Tidak Ada, Tidak Ada].

wumpus tidak bisa berada di [1,1], dengan aturan main, dan itu tidak bisa di [2,2] (atau agen akan mendeteksi bau busuk ketika itu di [2,1]). Oleh karena itu, agen dapat menyimpulkan bahwa wumpus ada di [1,3]. Notasi W! menunjukkan inferensi ini. Apalagi kekurangan angin sepoi-sepoi di [1,2] menyiratkan bahwa tidak ada lubang di [2,2]. Namun agen telah menyimpulkan bahwa harus ada jadilah [2,2] atau [3,1], jadi ini berarti harus dalam [3,1]. Ini cukup sulit kesimpulan, karena menggabungkan pengetahuan yang diperoleh pada waktu yang berbeda di tempat dan berbeda bergantung pada kurangnya persepsi untuk membuat satu langkah penting.

Halaman 7

Agen itu sekarang telah membuktikan pada dirinya sendiri bahwa tidak ada lubang atau wumpus di [2,2], jadi begitu tidak apa-apa untuk pindah ke sana. Kami tidak menunjukkan status pengetahuan agen di [2,2]; kami hanya berasumsi bahwa agen berbalik dan pindah ke [2,3], memberi kita Gambar 7.4 (b). Dalam [2,3], agen mendeteksi a glitter, jadi itu harus mengambil emas dan kemudian pulang.

Perhatikan bahwa dalam setiap kasus di mana agen menarik kesimpulan dari informasi yang tersedia pembentukan, kesimpulan itu *dijamin* benar jika informasi yang tersedia benar. Ini adalah sifat dasar penalaran logis. Di sisa bab ini, kami jelaskan bagaimana membangun agen logis yang dapat mewakili informasi dan menarik kesimpulan seperti itu dijelaskan dalam paragraf sebelumnya.

7.3 L OGIC

Bagian ini merangkum konsep dasar representasi logis dan penalaran. Ide-ide indah ini tidak tergantung pada bentuk logika apa pun. Oleh karena itu kami Selipkan rincian teknis dari formulir-formulir itu sampai bagian selanjutnya, dengan menggunakan yang sudah dikenal contoh aritmatika biasa.

Dalam Bagian 7.1, kami mengatakan bahwa basis pengetahuan terdiri dari kalimat. Kalimat-kalimat ini

SINTAKSIS diekspresikan sesuai dengan **sintaks** bahasa representasi, yang menentukan semua kalimat yang terbentuk dengan baik. Gagasan sintaks cukup jelas dalam aritmatika biasa: "X + y = 4" adalah kalimat yang dibentuk dengan baik, sedangkan "x4y + =" tidak.

SEMANTIK Logika juga harus mendefinisikan **semantik** atau makna kalimat. Semantik mendefinisikan

KEBENARAN yang **benar** dari setiap kalimat dengan masing-masing **dunia mungkin** . Misalnya, semantik

DUNIA YANG MUNGKIN untuk aritmatika menentukan bahwa kalimat "x + y = 4" adalah benar di dunia di mana x adalah 2 dan y adalah 2, tetapi salah di dunia di mana x adalah 1 dan y adalah 1. Dalam logika standar, setiap kalimat harus benar atau salah di setiap dunia yang memungkinkan — tidak ada "di antaranya." [1]

MODEL Ketika kita harus tepat, kita menggunakan istilah **model** di tempat "dunia yang mungkin." Sedangkan dunia yang mungkin dapat dianggap sebagai (berpotensi) lingkungan nyata dari agen tersebut mungkin atau mungkin tidak ada dalam, model adalah abstraksi matematis, yang masing-masing hanya memperbaiki kebenaran atau kepalsuan dari setiap kalimat yang relevan. Secara informal, kita mungkin memikirkan dunia yang mungkin sebagai, misalnya, memiliki x pria dan wanita duduk di meja bermain jembatan, dan kalimat x + y = 4 benar ketika ada empat orang secara total. Secara formal, model yang mungkin adalah adil semua kemungkinan penugasan bilangan real ke variabel x dan y. Setiap tugas perbaikan seperti itu kebenaran dari setiap kalimat aritmatika yang variabelnya adalah x dan y. Jika kalimat α benar dalam

KEPUASAN model m, kita mengatakan bahwa m **memenuhi** α atau kadang-kadang m **adalah model** α. Kami menggunakan notasi M (α) berarti himpunan semua model α.

Sekarang kami memiliki gagasan tentang kebenaran, kami siap untuk berbicara tentang alasan logis. Ini

HIBURAN melibatkan hubungan **entailment** logis antara kalimat-gagasan bahwa kalimat *mengikuti terendah secara logis* dari kalimat lain. Dalam notasi matematika, kita menulis

$$\alpha \models \beta$$

[1] **Logika fuzzy** , dibahas pada Bab 14, memungkinkan untuk tingkat kebenaran.

Halaman 8

(Sebuah) (b)

Gambar 7.5 Model yang mungkin untuk keberadaan lubang di kotak [1,2], [2,2], dan [3,1]. Itu KB yang sesuai dengan pengamatan tidak ada di [1,1] dan angin di [2,1] ditunjukkan oleh garis yang solid. (a) Garis putus-putus menunjukkan model α_1 (tidak ada lubang dalam [1,2]). (B) Garis putus-putus menunjukkan model α_2 (tidak ada lubang di [2,2]).

berarti kalimat α mencakup kalimat β. Definisi formal dari entailment adalah ini:
α | = β jika dan hanya jika, dalam setiap model di mana α benar, β juga benar. Menggunakan notasi saja diperkenalkan, kita bisa menulis

α | = β jika dan hanya jika M (α) ⊆ M (β).

(Perhatikan arah ⊆ di sini: jika α | = β, maka α adalah pernyataan yang *lebih kuat* dari β: itu mengesampingkan *lebih banyak* dunia yang mungkin.) Hubungan entailment akrab dari aritmatika; Kami bahagia dengan gagasan bahwa kalimat x = 0 mencakup kalimat xy = 0. Jelas, dalam model apa pun di mana x adalah nol, itu adalah kasus bahwa xy adalah nol (terlepas dari nilai y).

Kita bisa menerapkan jenis analisis yang sama dengan contoh penalaran dunia-wumpus yang diberikan di bagian sebelumnya. Pertimbangkan situasi pada Gambar 7.3 (b): agen telah mendeteksi tidak ada dalam [1,1] dan angin di [2,1]. Persepsi ini, dikombinasikan dengan pengetahuan agen aturan dunia wumpus, merupakan KB. Agen tertarik (antara lain hal-hal) apakah kotak yang berdekatan [1,2], [2,2], dan [3,1] berisi lubang. Masing-masing dari ketiganya kotak mungkin atau mungkin tidak mengandung lubang, jadi (untuk keperluan contoh ini) ada $2^3 = 8$ model yang mungkin. Kedelapan model ini ditunjukkan pada Gambar 7.5. [2]

KB dapat dianggap sebagai satu set kalimat atau sebagai satu kalimat yang menegaskan semua kalimat individu. KB salah dalam model yang bertentangan dengan apa yang diketahui agen—misalnya, KB salah dalam model mana pun [1,2] berisi lubang, karena ada tidak ada angin di [1,1]. Sebenarnya ada hanya tiga model di mana KB benar, dan ini

[2] Meskipun gambar menunjukkan model sebagai dunia wumpus parsial, mereka benar-benar tidak lebih dari tugas benar dan salah pada kalimat "ada lubang di [1,2]" dll. Model, dalam arti matematika, tidak perlu memiliki wumpus lapang 'orrible' di dalamnya.

Halaman 9

ditunjukkan dikelilingi oleh garis padat pada Gambar 7.5. Sekarang mari kita pertimbangkan dua kemungkinan kesimpulan:

α_1 = "Tidak ada lubang di [1,2]."
α_2 = "Tidak ada lubang di [2,2]."

Kami telah mengelilingi model α_1 dan α_2 dengan garis putus-putus pada Gambar 7.5 (a) dan 7.5 (b), masing-masing. Dengan inspeksi, kami melihat yang berikut:

di setiap model di mana KB benar, α_1 juga benar.

Karenanya, KB | = α_1 : tidak ada lubang di [1,2]. Kita juga bisa melihatnya

dalam beberapa model di mana KB benar, α_2 salah.

Karenanya, KB | = α 2 : agen *tidak dapat* menyimpulkan bahwa tidak ada lubang di [2,2]. (Juga tidak bisa menyimpulkan bahwa ada *adalah* lubang di [2,2].) 3

Contoh sebelumnya tidak hanya menggambarkan persyaratan tetapi juga menunjukkan bagaimana definisi entailment dapat diterapkan untuk memperoleh kesimpulan — yaitu, untuk melakukan **inferensi logis** .

INFERENSI LOGIS

PEMERIKSAAN MODEL

Algoritma inferensi yang diilustrasikan pada Gambar 7.5 disebut **pengecekan model** , karena memungkinkan menggabungkan semua model yang mungkin untuk memeriksa bahwa α benar di semua model di mana KB benar, yaitu, bahwa M (KB) ⊆ M (α).

Dalam memahami keterikatan dan kesimpulan, mungkin membantu untuk memikirkan himpunan semua konsekuensi quences dari KB sebagai tumpukan jerami dan dari α sebagai jarum. Entailment seperti jarum berada di dalam tumpukan jerami; kesimpulannya seperti menemukannya. Perbedaan ini diwujudkan dalam beberapa notasi formal: jika sebuah algoritma inferensi saya dapat memperoleh α dari KB, kami menulis

$$KB \vdash_i \alpha,$$

yang dilafalkan "α berasal dari KB oleh i" atau "i berasal α dari KB."

SUARA

MENJAGA KEBENARAN

Algoritma inferensi yang diturunkan hanya mencakup kalimat disebut **suara** atau **kebenaran-melestarikan** . Kesehatan adalah properti yang sangat diinginkan. Prosedur inferensi yang tidak sehat memperkirakan secara resmi mengada-ada seiring berjalannya waktu — itu mengumumkan penemuan jarum yang tidak ada. Sangat mudah untuk melihat bahwa memeriksa model, ketika itu berlaku, 4 adalah prosedur yang baik.

KELENGKAPAN

Properti **kelengkapan** juga diinginkan: algoritma inferensi lengkap jika itu dapat menurunkan kalimat apa pun yang disyaratkan. Untuk tumpukan jerami nyata, yang terbatas dalam batasnya, tampaknya jelas bahwa pemeriksaan sistematis selalu dapat memutuskan apakah jarum itu masuk tumpukan jerami. Untuk banyak basis pengetahuan, bagaimanapun, tumpukan konsekuensi tidak terbatas, dan kelengkapan menjadi masalah penting. 5 Untungnya, ada inferensi lengkap prosedur untuk logika yang cukup ekspresif untuk menangani banyak basis pengetahuan.

Kami telah menggambarkan proses penalaran yang kesimpulannya dijamin benar di dunia mana pun di mana premisnya benar; khususnya, *jika* KB *benar di dunia* nyata *, maka setiap kalimat* α yang *diturunkan dari* KB *dengan prosedur inferensi suara juga benar dalam real dunia.* Jadi, sementara proses inferensi beroperasi pada "sintaks" - konfigurasi fisik internal seperti bit dalam register atau pola blip listrik pada otak — prosesnya *sesuai*

[3] Agen dapat menghitung *probabilitas* bahwa ada lubang di [2,2]; Bab 13 menunjukkan caranya.

[4] Pengecekan model berfungsi jika ruang model terbatas — misalnya, di dunia wumpus dengan ukuran tetap. Untuk aritmatika, di sisi lain, ruang model tidak terbatas: bahkan jika kita membatasi diri ke bilangan bulat, di sana secara tak terhingga banyak pasangan nilai untuk x dan y dalam kalimat x + y = 4.

[5] Bandingkan dengan ruang pencarian tanpa batas di Bab 3, di mana pencarian kedalaman-pertama tidak lengkap.

Halaman 10

Gambar 7.6 Kalimat adalah konfigurasi fisik agen, dan penalaran adalah suatu proses membangun konfigurasi fisik baru dari yang lama. Penalaran logis harus yakin bahwa konfigurasi baru mewakili aspek dunia yang sebenarnya mengikuti dari aspek yang mewakili konfigurasi lama.

untuk hubungan dunia nyata di mana beberapa aspek dari dunia nyata adalah kasus 6 berdasarkan aspek lain dari dunia nyata menjadi kasusnya. Korespondensi ini antara dunia dan representasi diilustrasikan pada Gambar 7.6.

GROUNDING

Masalah terakhir yang harus dipertimbangkan adalah **landasan** — hubungan antara penalaran logis proses dan lingkungan nyata di mana agen itu ada. Secara khusus, *bagaimana kita tahu bahwa* KB *benar di dunia nyata?* (Bagaimanapun, KB hanya "sintaks" di dalam kepala agen.) Ini adalah pertanyaan filosofis tentang mana banyak, banyak buku telah ditulis. (Lihat Bab 26.) Jawaban sederhana adalah bahwa sensor agen membuat koneksi. Sebagai contoh,

agen dunia wumpus kami memiliki sensor bau. Program agen menciptakan kalimat yang cocok setiap kali ada bau. Lalu, setiap kali kalimat itu ada di basis pengetahuan, itu benar di dunia nyata. Dengan demikian, makna dan kebenaran kalimat persepsi didefinisikan oleh proses penginderaan dan konstruksi kalimat yang menghasilkannya. Bagaimana dengan sisa pengetahuan agen, seperti keyakinannya bahwa wumpus menyebabkan bau di kotak yang berdekatan? Ini bukan representasi langsung dari satu persepsi, tetapi aturan umum — diturunkan, mungkin, dari pengalaman perseptual tetapi tidak identik dengan pernyataan pengalaman itu. Aturan umum seperti ini dihasilkan oleh proses konstruksi kalimat yang disebut **belajar** , yang merupakan subjek Bagian V. Belajar itu salah. Bisa jadi itu kasus wumpus yang menyebabkan bau *kecuali pada 29 Februari di tahun kabisat* , yaitu saat mereka mandi. Jadi, KB mungkin tidak benar dalam dunia nyata, tetapi dengan prosedur pembelajaran yang baik, ada alasan untuk optimis.

7.4 P ROPOSITIONAL L OGIC : AV ERY S imple L OGIC

PROPOSITIONAL LOGIKA

Kami sekarang menyajikan logika sederhana namun kuat yang disebut **logika proposisional** . Kami membahas sintaksisnya logika proposisional dan semantiknya - cara di mana kebenaran kalimat ditentukan beranjau. Kemudian kita melihat **entailment** — hubungan antara kalimat dan kalimat lain yang mengikutinya — dan lihat bagaimana ini mengarah pada algoritma sederhana untuk inferensi logis. Everything terjadi, tentu saja, di dunia wumpus.

[6] Seperti yang dikatakan Wittgenstein (1922) dalam *Tractatus* -nya yang terkenal : "Dunia adalah segalanya yang ada."

Halaman 11

7.4.1 Sintaks

KALIMAT ATOMIK

DALIL SIMBOL

The **sintaks** logika proposisional mendefinisikan kalimat yang diijinkan. The **kalimat atom** terdiri dari **simbol proposisi** tunggal . Setiap simbol tersebut mewakili proposisi yang bisa benar atau salah. Kami menggunakan simbol yang dimulai dengan huruf besar dan mungkin mengandung lainnya huruf atau subskrip, misalnya: P, Q, R, $W_{1,3}$ dan North. Nama-nama itu arbitrer tetapi sering dipilih untuk memiliki nilai mnemonik — kami menggunakan $W_{1,3}$ untuk mendukung proposisi bahwa wumpus ada di [1,3]. (Ingat bahwa simbol seperti $W_{1,3}$ adalah *atom* , yaitu, W, 1, dan 3 bukan bagian simbol yang berarti.) Ada dua simbol proposisi dengan tetap arti: Benar adalah proposisi selalu-benar dan Salah adalah proposisi selalu-salah.

KOMPLEKS KALIMAT LOGIS SAMBUNGAN

Kalimat kompleks dibangun dari kalimat yang lebih sederhana, menggunakan tanda kurung dan **logis penghubung** . Ada lima penghubung yang umum digunakan:

PENYANGKALAN

HARFIAH

¬ (tidak). Kalimat seperti $W_{1,3}$ disebut **negasi** $W_{1,3}$. Sebuah **literal** adalah baik sebuah kalimat atom (**literal positif**) atau kalimat atom yang dinegasikan (**literal negatif**).

KONJUNGSI

∧ (dan). Sebuah kalimat yang konektif utamanya adalah ∧, seperti $W_{1,3} \wedge P_{3,1}$, disebut **con persimpangan** ; bagian-bagiannya adalah **konjungsi** . (∧ terlihat seperti "A" untuk "Dan.")

PEMISAHAN

∨ (atau). Sebuah kalimat menggunakan ∨, seperti $(W_{1,3} \wedge P_{3,1}) \vee W_{2,2}$, adalah **disjungsi** dari **disjuncts** $(W_{1,3} \wedge P_{3,1})$ dan $W_{2,2}$. (Secara historis, ∨ berasal dari bahasa Latin "vel," yang artinya "Atau." Bagi kebanyakan orang, lebih mudah untuk mengingat ∨ sebagai terbalik ∧.)

IMPLIKASI

PREMISE

KESIMPULAN

ATURAN

⇒ (tersirat). Kalimat seperti $(W_{1,3} \wedge P_{3,1}) \Rightarrow \neg W_{2,2}$ disebut **implikasi** (atau disional). Its **premis** atau **yg** adalah $(W_{1,3} \wedge P_{3,1})$, dan yang **kesimpulan** atau **akibat** adalah $\neg W_{2,2}$. Implikasi juga dikenal sebagai **aturan** atau pernyataan **if-then** . Implikasi simbol kadang-kadang ditulis dalam buku-buku lain sebagai ⊃ atau →.

BICONDITIONAL

⇔ (jika dan hanya jika). Kalimat $W_{1,3} \text{ WW}_{2,2}$ adalah **bikondisional** . Beberapa buku lain tulis ini sebagai ≡.

Kalimat → AtomicSentence | Rasa Kompleks

AtomicSentence → True | Salah | P | Q | R | ...

ComplexSentence → **(** Kalimat **)** | **[** Kalimat **]**
| ¬ Kalimat
| Kalimat ∧ Kalimat

| Kalimat ∨ Kalimat
| Kalimat ⇒ Kalimat
| Kalimat ⇔ Kalimat

O PERATOR P RECEDENCE : ¬, ∧, ∨, ⇒, ⇔

Gambar 7.7 Tata bahasa BNF (Backus – Naur Form) kalimat dalam logika proposisional, bersama dengan prioritas operator, dari tertinggi ke terendah.

Halaman 12

Gambar 7.7 memberikan tata bahasa formal dari logika proposisional; lihat halaman 1060 jika tidak akrab dengan notasi BNF. Tata bahasa BNF dengan sendirinya bersifat ambigu; sebuah kalimat dengan beberapa operator dapat diurai oleh tata bahasa dalam berbagai cara. Untuk menghilangkan ambiguitas kami mendefinisikan prioritas untuk setiap operator. Operator "tidak" (¬) memiliki prioritas tertinggi, yang berarti bahwa dalam kalimat ¬A ∧ B ¬ mengikat paling erat, memberi kita yang setara dari (¬ A) ∧B daripada ¬ (A∧B). (Notasi untuk aritmatika biasa adalah sama: −2 + 4 adalah 2, bukan –6.) Ketika ragu, gunakan tanda kurung untuk memastikan interpretasi yang benar. Kotak tanda kurung sama dengan tanda kurung; pilihan kurung kotak atau tanda kurung adalah semata-mata untuk memudahkan manusia membaca kalimat.

7.4.2 Semantik

Setelah menentukan sintaksis logika proposisional, sekarang kami menentukan semantiknya. Se mantika mendefinisikan aturan untuk menentukan kebenaran kalimat sehubungan dengan tertentu model. Dalam logika proposisional, model hanya memperbaiki nilai **kebenaran** — **benar** atau salah — untuk simbol usulan ery. Misalnya, jika kalimat dalam basis pengetahuan menggunakan kata simbol proposisi $P_{1,2}$, $P_{2,2}$, dan $P_{3,1}$, maka satu model yang mungkin adalah

NILAI KEBENARAN

$$m_1 = \{P_{1,2} = \text{false}, P_{2,2} = \text{false}, P_{3,1} = \text{true}\}.$$

Dengan tiga simbol proposisi, ada $2^3 = 8$ model yang mungkin — persis yang digambarkan pada Gambar 7.5. Perhatikan, bagaimanapun, bahwa model adalah objek matematika murni tanpa koneksi yang diperlukan untuk dunia wumpus. $P_{1,2}$ hanyalah sebuah simbol; itu mungkin berarti “ada lubang di [1,2] ”atau“ Saya di Paris hari ini dan besok. ”

Semantik untuk logika proposisional harus menentukan cara menghitung nilai kebenaran kalimat *apa pun* , diberi model. Ini dilakukan secara rekursif. Semua kalimat dibangun dari kalimat atom dan lima penghubung; Oleh karena itu, kita perlu menentukan cara menghitung kebenaran kalimat atom dan bagaimana menghitung kebenaran kalimat yang dibentuk dengan masing - masing lima penghubung. Kalimat atom mudah:

- Benar adalah benar dalam setiap model dan Salah adalah salah dalam setiap model.
- Nilai kebenaran dari setiap simbol proposisi lainnya harus ditentukan secara langsung dalam model. Misalnya, dalam model $m_{1\ yang}$ diberikan sebelumnya, $P_{1,2}$ salah.

Untuk kalimat yang kompleks, kami memiliki lima aturan, yang berlaku untuk setiap subsensi P dan Q dalam setiap model m (di sini "iff" berarti "jika dan hanya jika"):

- ¬P benar jika P salah dalam m.
- P ∧ Q benar jika kedua P dan Q benar dalam m.
- P ∨ Q benar jika f atau P benar dalam m.
- P ⇒ Q benar kecuali P benar dan Q salah dalam m.
- P ⇔ Q benar jika P dan Q keduanya benar atau keduanya salah dalam m.

MEJA KEBENARAN

Aturan juga dapat diekspresikan dengan **tabel kebenaran** yang menentukan nilai kebenaran kompleks kalimat untuk setiap kemungkinan penugasan nilai kebenaran ke komponen-komponennya. Tabel kebenaran untuk lima penghubung diberikan pada Gambar 7.8. Dari tabel ini, nilai kebenaran dari setiap kalimat s dapat dihitung sehubungan dengan model apa pun dengan evaluasi rekursif sederhana. Sebagai contoh,

Halaman 13

P	Q	¬ P.	P ∧ Q	P ∨ Q	P ⇒ Q	P ⇔ Q
Salah	Salah	benar	Salah	Salah	benar	benar
Salah	benar	benar	Salah	benar	benar	Salah
benar	Salah	Salah	Salah	benar	Salah	Salah
benar	benar	Salah	benar	benar	benar	benar

Gambar 7.8 Tabel kebenaran untuk lima penghubung logis. Untuk menggunakan tabel untuk menghitung, untuk Sebagai contoh, nilai P ∨ Q ketika P benar dan Q salah, lihat pertama di sebelah kiri untuk baris di mana P benar dan Q salah (baris ketiga). Kemudian lihat di baris itu di bawah kolom P ∨Q untuk melihat hasilnya: true.

kalimat $\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1})$, dievaluasi dalam m_1, memberikan true ∧ (false ∨ true) = true ∧ true = benar. Latihan 7.3 meminta Anda untuk menulis algoritma PL-TRUE? (S, m), yang menghitung nilai kebenaran kalimat logika proposisional dalam model m.

Tabel kebenaran untuk "dan," "atau," dan "tidak" sesuai dengan intuisi kita tentang kata-kata bahasa Inggris. Poin utama dari kemungkinan kebingungan adalah bahwa P ∨ Q benar ketika P benar atau Q benar *atau keduanya* . Penghubung yang berbeda, yang disebut "eksklusif atau" ("singkatnya"), menghasilkan salah ketika kedua pemisahan benar. [7] Tidak ada konsensus pada simbol untuk eksklusif atau; beberapa pilihan adalah ˙∨ atau = atau ⊕.

Tabel kebenaran untuk ⇒ mungkin tidak cukup sesuai dengan pemahaman intuitif seseorang tentang "P menyiratkan Q" atau "jika P maka Q." Untuk satu hal, logika proposisional tidak memerlukan hubungan *sebab akibat* atau *relevansi* antara P dan Q. Kalimat "5 aneh menyiratkan Tokyo adalah ibu kota Jepang" adalah kalimat yang benar dari logika proposisional (di bawah interpretasi normal), meskipun demikian kalimat bahasa Inggris yang jelas aneh. Hal lain yang membingungkan adalah implikasinya benar setiap kali pendahulunya salah. Misalnya, "5 bahkan menyiratkan Sam pintar" adalah benar, terlepas dari apakah Sam itu pintar. Ini tampaknya aneh, tetapi masuk akal jika Anda memikirkannya "P ⇒ Q" mengatakan, "Jika P benar, maka saya mengklaim bahwa Q benar. Kalau tidak, aku yang membuat no claim. "Satu-satunya cara agar kalimat ini *salah* adalah jika P benar tetapi Q salah.

Bikondisional, P ⇔ Q, benar setiap kali P ⇒ Q dan Q ⇒ P benar. Di Bahasa Inggris, ini sering ditulis sebagai "P if and only if Q." Banyak aturan dunia wumpus paling baik ditulis menggunakan ⇔. Sebagai contoh, sebuah kotak adalah semilir *jika* sebuah kotak tetangga memiliki lubang, dan bujur sangkar adalah semilir *hanya jika* sebuah alun-alun tetangga memiliki sebuah lubang. Jadi kita perlu bikondisional,

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}),$$

di mana $B_{1,1}$ berarti ada angin di [1,1].

7.4.3 Basis pengetahuan sederhana

Sekarang kita telah mendefinisikan semantik untuk logika proposisional, kita dapat membangun pengetahuan dasar untuk dunia wumpus. Kami fokus pertama pada aspek *abadi* dari dunia wumpus, meninggalkan aspek yang bisa berubah untuk bagian selanjutnya. Untuk saat ini, kita memerlukan simbol berikut untuk setiap lokasi [x, y]:

[7] Latin memiliki kata yang terpisah, *aut* , untuk eksklusif atau.

Halaman 14

$P_{x,y}$ benar jika ada lubang di [x, y].
$W_{x,y}$ benar jika ada wumpus di [x, y], hidup atau mati.
$B_{x,y}$ benar jika agen merasakan angin di [x, y].
$S_{x,y}$ benar jika agen merasakan bau pada [x, y].

Kalimat yang kita tulis akan cukup untuk memperoleh $\neg P_{1,2}$ (tidak ada lubang dalam [1,2]), seperti yang dilakukan secara informal di Bagian 7.3. Kami memberi label pada setiap kalimat R_i sehingga kami dapat merujuknya:

• Tidak ada lubang di [1,1]:

$$R_1 : \neg P_{1,1} .$$

• Sebuah kotak berangin jika dan hanya jika ada sebuah lubang di sebuah kotak tetangga. Ini pasti dinyatakan untuk setiap kotak; untuk saat ini, kami hanya menyertakan kotak yang relevan:

$$R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}).$$
$$R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1}).$$

• Kalimat sebelumnya benar di semua dunia wumpus. Sekarang kami menyertakan angin persepsi untuk dua kotak pertama yang dikunjungi di dunia spesifik agen, memimpin untuk situasi pada Gambar 7.3 (b).

$$R_4 : \neg B_{1,1} .$$
$$R_5 : B_{2,1} .$$

7.4.4 Prosedur inferensi sederhana

Tujuan kami sekarang adalah untuk memutuskan apakah KB | = α untuk beberapa kalimat α. Misalnya, adalah $\neg P_{1,2}$ disyaratkan oleh KB kami? Algoritma pertama kami untuk inferensi adalah pendekatan pengecekan model yaitu a implementasi langsung dari definisi entailment: sebutkan model, dan periksa itu α benar di setiap model di mana KB benar. Model adalah penugasan benar atau salah setiap simbol proposisi. Kembali ke contoh dunia wumpus kami, proposisi yang relevan simbol tion adalah $B_{1,1}$, $B_{2,1}$, $P_{1,1}$, $P_{1,2}$, $P_{2,1}$, $P_{2,2}$, dan $P_{3,1}$. Dengan tujuh simbol, ada $2^7 = 128$ model yang mungkin; dalam tiga hal ini, KB benar (Gambar 7.9). Dalam tiga model itu, $\neg P_{1,2}$ benar, maka tidak ada lubang dalam [1,2]. Di sisi lain, $P_{2,2}$ benar dalam dua dari tiga model dan false dalam satu, jadi kami belum bisa mengatakan apakah ada lubang di [2,2].

Gambar 7.9 mereproduksi dalam bentuk yang lebih tepat alasan yang diilustrasikan pada Gambar 7.5. SEBUAH Algoritma umum untuk memutuskan entailment dalam logika proposisional ditunjukkan pada Gambar 7.10. Suka algoritma B ACKTRACKING -S EARCH di halaman 215, TT-E NTAILS ? melakukan rekursif enumerasi ruang tugas terbatas hingga simbol. Algoritma ini **baik** karena itu mengimplementasikan langsung definisi entailment, dan **menyelesaikan** karena ia berfungsi untuk setiap KB dan α dan selalu berakhir — hanya ada banyak model untuk diperiksa.

Tentu saja, "banyak sekali" tidak selalu sama dengan "sedikit." Jika KB dan α mengandung n simbol dalam semua, maka ada model 2^n . Dengan demikian, kompleksitas waktu dari algoritma ini adalah $O(2^n)$. (Kompleksitas ruang hanya O (n) karena enumerasi adalah kedalaman-pertama.) Kemudian bab ini kami menunjukkan algoritma yang jauh lebih efisien dalam banyak kasus. Sayangnya, pengajuan proposisional adalah melengkapi NP bersama (yaitu, mungkin tidak lebih mudah dari melengkapi NP — lihat Lampiran A), sehingga *setiap algoritma inferensi yang diketahui untuk logika proposisional memiliki kasus terburuk kompleksitas yang eksponensial dalam ukuran input.*

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	false	false	false	false	false	false	benar	benar	benar	benar	salah	Salah
false	false	false	false	false	false	benar	benar	benar	salah	benar	salah	Salah
...	...	...	...	...	...	...	...	...	...	...	...	...
false	true	false	false	false	false	false	benar	benar	salah	benar	benar	Salah
false	true	false	false	false	false	benar	benar	benar	benar	benar	benar	benar
false	true	false	false	salah	benar	salah	benar	benar	benar	benar	benar	benar
false	true	false	false	false	benar	benar	benar	benar	benar	benar	benar	benar

salah	benar	salah	salah	benar	salah	salah			benar	salah	benar	benar	benar	Salah
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
benar	benar	benar	benar	benar	benar	benar	salah	benar	benar	salah	benar			Salah

Gambar 7.9 Tabel kebenaran dibangun untuk basis pengetahuan yang diberikan dalam teks. *KB* benar jika R_1 hingga R_5 benar, yang terjadi hanya dalam 3 dari 128 baris (yang digarisbawahi dalam kolom sebelah kanan). Dalam semua 3 baris, $P_{1,2}$ salah, sehingga tidak ada lubang di [1,2]. Di samping itu, mungkin ada (atau mungkin tidak) lubang di [2,2].

fungsi TT-ENTAILS? (KB, α) **mengembalikan** benar atau salah
  input : KB, basis pengetahuan, kalimat dalam logika proposisional
    α, kueri, kalimat dalam logika proposisional

  simbol ← daftar simbol proposisi dalam KB dan α
  return TT-CHECK-ALL (KB, α, simbol, {})

fungsi TT-CHECK-ALL (KB, α, simbol, model) **mengembalikan** benar atau salah
  jika EMPTY? (simbol) **lalu**
    jika PL-TRUE? (KB, model) **lalu kembalikan** PL-TRUE? (α, model)
    lain mengembalikan benar // *ketika KB salah, selalu kembali benar*
  lain lakukan
    P ← FIRST (simbol)
    sisanya ← REST (simbol)
    return (TT-CHECK-ALL (KB, α, istirahat, model ∪ {P = true})
      dan
      TT-CHECK-ALL (KB, α, istirahat, model ∪ {P = false}))

Gambar 7.10 Algoritme enumerasi tabel kebenaran untuk memutuskan pengunduran diri proposisional. (TT singkatan dari tabel kebenaran.) PL-TRUE? mengembalikan *true* jika kalimat berlaku dalam model. Itu *model* variabel mewakili *model* parsial — penugasan untuk beberapa simbol. Kunci-kata " **and** " digunakan di sini sebagai operasi logis pada dua argumennya, menghasilkan benar atau salah.

Halaman 16

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$ komutatif $\wedge$
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$ komutatif $\vee$
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$ asosiatif $\wedge$
$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$ asosiatif $\vee$
$\neg(\neg\alpha) \equiv \alpha$ penghapusan negasi ganda
$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ kontraposisi
$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ eliminasi implikasi
$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$ eliminasi bikondisional
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ De Morgan
$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ De Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ distributivitas $\wedge$ lebih dari $\vee$
$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$ distributivitas $\vee$ lebih dari $\wedge$

Gambar 7.11 Kesetaraan logis standar. Simbol α, β, dan γ berarti arbitrary kalimat-kalimat logika proposisional.

7.5 PROPOSITIONAL THEOREM PROVING

PENYEDIAAN TEORI

Sejauh ini, kami telah menunjukkan cara menentukan entailment dengan *memeriksa model* : menghitung model dan menunjukkan bahwa kalimat itu harus ada di semua model. Di bagian ini, kami menunjukkan bagaimana ment dapat dilakukan dengan **pembuktian teorema —** menerapkan aturan inferensi langsung ke kalimat

di basis pengetahuan kami untuk membangun bukti dari kalimat yang diinginkan tanpa model konsultasi. Jika jumlah model besar tetapi panjang buktinya pendek, maka teorema membuktikan bisa lebih efisien daripada pengecekan model.

Sebelum kita terjun ke detail algoritma pembuktian teorema, kita perlu beberapa konsep tambahan terkait dengan entailment. Konsep pertama adalah **kesetaraan logis** : dua tences α dan β secara logis setara jika benar dalam set model yang sama. Kami menulis ini sebagai α ≡ β. Sebagai contoh, kita dapat dengan mudah menunjukkan (menggunakan tabel kebenaran) bahwa P ∧ Q dan Q ∧ P secara logis setara; kesetaraan lainnya ditunjukkan pada Gambar 7.11. Kesetaraan ini memainkan banyak peran yang sama dalam logika seperti halnya identitas aritmatika dalam matematika biasa. Sebuah definisi alternatif kesetaraan adalah sebagai berikut: setiap dua kalimat α dan β adalah setara hanya jika masing-masing memerlukan yang lain:

LOGIS PERSAMAAN DERAJATNYA

$$\alpha \equiv \beta \text{ jika dan hanya jika } \alpha \models \beta \text{ dan } \beta \models \alpha.$$

KEABSAHAN

Konsep kedua yang kita butuhkan adalah **validitas** . Sebuah kalimat valid jika itu benar di *semua* model. Untuk

ULANGAN YG TDK BERGUNA

Misalnya, kalimat P ∨ ¬P valid. Kalimat yang valid juga dikenal sebagai **tautologi** —mereka yang *tentu* benar. Karena kalimat True itu benar di semua model, setiap kalimat valid secara logis setara dengan True. Apa bagusnya kalimat yang valid? Dari definisi kami tentang

DEDUKSI DALIL

mensyaratkan, kita dapat memperoleh **teorema deduksi** , yang dikenal oleh orang Yunani kuno:

Untuk setiap kalimat α *dan* β , $\alpha \models \beta$ *jika dan hanya jika kalimat* $(\alpha \Rightarrow \beta)$ *valid.*

(Latihan 7.5 meminta bukti.) Oleh karena itu, kita dapat memutuskan apakah $\alpha \models \beta$ dengan memeriksa bahwa $(\alpha \Rightarrow \beta)$ adalah benar dalam setiap model — yang pada dasarnya adalah apa yang dilakukan algoritma inferensi pada Gambar 7.10—

Halaman 17

atau dengan membuktikan bahwa $(\alpha \Rightarrow \beta)$ setara dengan True. Sebaliknya, teorema deduksi menyatakan bahwa setiap kalimat implikasi yang valid menggambarkan inferensi yang sah.

KEPUASAN

Konsep terakhir yang kita butuhkan adalah **kepuasan** . Sebuah kalimat memuaskan jika itu benar dalam, atau puas dengan, *beberapa* model. Misalnya, basis pengetahuan yang diberikan sebelumnya, $(R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5)$, cukup memuaskan karena ada tiga model yang benar, seperti yang ditunjukkan pada Gambar 7.9. Kepuasan dapat diperiksa dengan menghitung model yang mungkin sampai satu ditemukan yang memenuhi hukuman. Masalah menentukan kepuasan kalimat

DUDUK

dalam logika proposisional — masalah **SAT** — adalah masalah pertama yang terbukti lengkap NP. Banyak masalah dalam ilmu komputer adalah masalah yang benar-benar memuaskan. Misalnya, semua kendala masalah kepuasan dalam Bab 6 tanyakan apakah kendala dapat dipenuhi oleh beberapa tugas.

Validitas dan kepuasan tentu saja terhubung: α valid jika ¬α tidak memuaskan; sebaliknya, α memuaskan jika ¬α tidak valid. Kami juga memiliki hasil bermanfaat berikut:

$\alpha \models \beta$ *jika dan hanya jika kalimat* $(\alpha \wedge \neg\beta)$ *tidak memuaskan.*

Membuktikan β dari α dengan memeriksa ketidakpuasan $(\alpha \wedge \neg\beta)$ sesuai persis dengan

REDUKSI AD ABSURDUM

teknik pembuktian matematis standar *reductio ad absurdum* (secara harfiah, "reduksi menjadi

SANGGAHAN

hal yang absurd "). Ini juga disebut bukti oleh **sangkalan** atau bukti oleh **kontradiksi** . Seseorang mengasumsikan a

KONTRADIKSI

kalimat β menjadi salah dan menunjukkan bahwa ini mengarah pada kontradiksi dengan aksioma yang diketahui α. Ini kontradiksi adalah apa yang dimaksud dengan mengatakan bahwa kalimat $(\alpha \wedge \neg\beta)$ tidak memuaskan.

7.5.1 Inferensi dan bukti

ATURAN INFERENSI

Bagian ini mencakup **aturan inferensi** yang dapat diterapkan untuk memperoleh **bukti** — rantai kesimpulan.

BUKTI

Sions yang mengarah ke tujuan yang diinginkan. Aturan paling terkenal disebut **Modus Ponens** (bahasa Latin untuk

MODUS PONENS

mode yang menegaskan) dan ditulis

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}.$$

Notasi berarti bahwa, setiap kali kalimat dalam bentuk $\alpha \Rightarrow \beta$ dan α diberikan, maka kalimat β dapat disimpulkan. Misalnya, jika (WumpusAhead ∧WumpusAlive) ⇒ Tembak dan (WumpusAhead ∧ WumpusAlive) diberikan, maka Tembak dapat disimpulkan.

DAN-PENGHAPUSAN

Aturan inferensi lain yang bermanfaat adalah **Dan-Eliminasi** , yang mengatakan bahwa, dari kata sambung, salah satu konjungsi dapat disimpulkan:

$$\frac{\alpha \wedge \beta}{\alpha}.$$

Misalnya, dari (WumpusAhead ∧ WumpusAlive), WumpusAlive dapat disimpulkan.

Dengan mempertimbangkan nilai kebenaran yang mungkin dari α dan β, orang dapat dengan mudah menunjukkan Modus itu Ponens dan Dan-Eliminasi adalah suara sekali dan untuk semua. Aturan-aturan ini kemudian dapat digunakan di setiap contoh khusus di mana mereka berlaku, menghasilkan kesimpulan suara tanpa perlu menghitung model.

Semua kesetaraan logis pada Gambar 7.11 dapat digunakan sebagai aturan inferensi. Untuk ujian-ple, kesetaraan untuk eliminasi bikondisional menghasilkan dua aturan inferensi

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)} \quad \text{dan} \quad \frac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}.$$

Halaman 18

Tidak semua aturan inferensi bekerja di kedua arah seperti ini. Misalnya, kita tidak dapat menjalankan Modus Ponens dalam arah yang berlawanan untuk mendapatkan $\alpha \Rightarrow \beta$ dan α dari β.

Mari kita lihat bagaimana aturan inferensi dan persamaan ini dapat digunakan di dunia wumpus. Kami mulai dengan basis pengetahuan yang mengandung R_1 hingga R_5 dan menunjukkan bagaimana membuktikan $\neg P_{1,2}$, artinya, tidak ada lubang dalam [1,2]. Pertama, kami menerapkan eliminasi bikondisional pada R_2 untuk mendapatkan

$$R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}).$$

Kemudian kita menerapkan Dan-Eliminasi ke R_6 untuk mendapatkan

$$R_7 : ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}).$$

Kesetaraan logis untuk kontrasepsi memberi

$$R_8 : (\neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1})).$$

Sekarang kita dapat menerapkan Modus Ponens dengan R_8 dan percept R_4 (yaitu, $\neg B_{1,1}$), untuk mendapatkan

$$R_9 : \neg (P_{1,2} \vee P_{2,1}).$$

Akhirnya, kami menerapkan aturan De Morgan, memberikan kesimpulan

$$R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}.$$

Artinya, [1,2] atau [2,1] tidak mengandung lubang.

Kami menemukan bukti ini dengan tangan, tetapi kami dapat menerapkan algoritma pencarian apa pun di Bab 3 untuk menemukan urutan langkah-langkah yang merupakan bukti. Kami hanya perlu mendefinisikan masalah bukti sebagai berikut:

- I NITIAL S TATE : basis pengetahuan awal.
- A CTIONS : serangkaian tindakan terdiri dari semua aturan inferensi yang diterapkan untuk semua rumah yang cocok dengan bagian atas dari aturan inferensi.
- R ESULT : hasil dari suatu tindakan adalah menambahkan kalimat di bagian bawah kesimpulan aturan.
- G OAL : tujuannya adalah keadaan yang mengandung kalimat yang kami coba buktikan.

Jadi, mencari bukti adalah alternatif untuk menghitung model. Dalam banyak kasus praktis *menemukan bukti dapat lebih efisien karena bukti dapat mengabaikan proposisi yang tidak relevan, tidak berapapun jumlah mereka.* Misalnya, bukti yang diberikan sebelumnya mengarah ke $\neg P_{1,2} \wedge \neg P_{2,1}$ tidak menyebutkan proposisi $B_{2,1}$, $P_{1,1}$, $P_{2,2}$, atau $P_{3,1}$. Mereka bisa diabaikan karena proposisi sasaran, $P_{1,2}$, hanya muncul dalam kalimat R_2; proposisi lain dalam R_2 hanya muncul di R_4 dan R_2; jadi R_1, R_3, dan R_5 tidak ada hubungannya dengan buktinya. Hal yang sama juga akan terjadi tahan bahkan jika kita menambahkan satu juta kalimat lagi ke basis pengetahuan; tabel kebenaran sederhana Algoritma, di sisi lain, akan kewalahan oleh ledakan eksponensial model.

MONOTONITAS

Satu sifat terakhir dari sistem logis adalah **monotonisitas**, yang mengatakan bahwa himpunan en kalimat berekor hanya dapat *meningkat* karena informasi ditambahkan ke basis pengetahuan. [8] Untuk apa saja kalimat α dan β,

$$\text{jika } KB \models \alpha \text{ maka } KB \wedge \beta \models \alpha.$$

[8] Logika **nonmonotonik**, yang melanggar sifat monotonisitas, menangkap sifat umum dari sifat manusia. soning: mengubah pikiran seseorang. Mereka dibahas dalam Bagian 12.6.

Halaman 19

Sebagai contoh, misalkan basis pengetahuan berisi pernyataan tambahan β yang menyatakan bahwa ada persis delapan lubang di dunia. Pengetahuan ini mungkin membantu agen menarik *tambahan* con clusions, tetapi tidak dapat membatalkan kesimpulan α yang sudah disimpulkan — seperti kesimpulan bahwa tidak ada lubang di [1,2]. Monotonicity berarti aturan inferensi dapat diterapkan kapan saja Tempat-tempat yang cocok ditemukan dalam basis pengetahuan — kesimpulan dari aturan harus mengikuti *terlepas dari apa lagi yang ada di basis pengetahuan* .

7.5.2 Bukti dengan resolusi

Kami berpendapat bahwa aturan inferensi yang dicakup sejauh ini adalah *baik* , tetapi kami belum membahas pertanyaan tentang *kelengkapan* untuk algoritma inferensi yang menggunakannya. Algoritma pencarian seperti pencarian pendalaman berulang (halaman 89) lengkap dalam arti yang akan mereka temukan tujuan apa pun yang dapat dijangkau, tetapi jika aturan inferensi yang tersedia tidak memadai, maka tujuannya tidak dapat dijangkau — tidak ada bukti yang hanya menggunakan aturan inferensi tersebut. Misalnya, jika kita dihapus aturan eliminasi bikondisional, bukti di bagian sebelumnya tidak akan melewati.
Bagian saat ini memperkenalkan aturan inferensi tunggal, **resolusi** , yang menghasilkan lengkap algoritma inferensi ketika digabungkan dengan algoritma pencarian lengkap.

Kita mulai dengan menggunakan versi sederhana dari aturan resolusi di dunia wumpus. Biarkan kami pertimbangkan langkah-langkah menjelang Gambar 7.4 (a): agen kembali dari [2,1] ke [1,1] lalu pergi ke [1,2], di mana ia merasakan bau, tetapi tidak ada angin. Kami menambahkan fakta berikut ke dasar pengetahuan:

$$R_{11} : \quad \neg B_{1,2} .$$
$$R_{12} : \quad B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3}) .$$

Dengan proses yang sama yang mengarah ke R_{10} sebelumnya, kita sekarang bisa mendapatkan tidak adanya lubang di [2,2] dan [1,3] (ingat bahwa [1,1] sudah dikenal tanpa pitless):

$$R_{13} : \quad \neg P_{2,2} .$$
$$R_{14} : \quad \neg P_{1,3} .$$

Kita juga bisa menerapkan eliminasi bikondisional ke R_3 , diikuti oleh Modus Ponens dengan R_5 , ke dapatkan fakta bahwa ada lubang di [1,1], [2,2], atau [3,1]:

$$R_{15} : \quad P_{1,1} \vee P_{2,2} \vee P_{3,1} .$$

Sekarang datang aplikasi pertama dari aturan resolusi: literal $\neg P_{2,2}$ dalam R_{13} diselesaikan *dengan* $P_{2,2}$ literal dalam R_{15} untuk memberikan **resolusi**

OBAT ANTI RADANG

$$R_{16} : \quad P_{1,1} \vee P_{3,1} .$$

Dalam Bahasa Inggris; jika ada lubang di salah satu [1,1], [2,2], dan [3,1] dan tidak ada di [2,2], maka itu ada di [1,1] atau [3,1]. Demikian pula, literal $\neg p_{1,1}$ di R_1 menyelesaikan dengan P literal $_{1,1}$ di R_{16} untuk memberikan

$$R_{17} : \quad P_{3,1} .$$

Dalam bahasa Inggris: jika ada lubang di [1,1] atau [3,1] dan tidak ada di [1,1], maka ada di [3,1]. Ini yang terakhir dua langkah inferensi adalah contoh aturan inferensi **resolusi unit** ,

RESOLUSI UNIT

$$\frac{l_1 \vee \cdots \vee l_k , \qquad m}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k} ,$$

KOMPLEMENTER LITERAL

di mana masing-masing l adalah literal dan l_i dan m adalah **literal komplementer** (yaitu, satu adalah negasi

Halaman 20

AYAT

dari yang lain). Dengan demikian, aturan resolusi satuan mengambil **klausa** — disjungsi literal — dan a literal dan menghasilkan klausa baru. Perhatikan bahwa satu literal dapat dilihat sebagai disjungsi dari satu literal, juga dikenal sebagai **klausa satuan** .

KLAUS SATUAN

RESOLUSI

Aturan resolusi unit dapat digeneralisasi ke aturan **resolusi** penuh ,

$$\frac{l_1 \vee \cdots \vee l_k, \quad m_1 \vee \cdots \vee m_n}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n},$$

di mana l_i dan m_j adalah literal melengkapi. Ini mengatakan bahwa resolusi membutuhkan dua klausa dan menghasilkan klausa baru yang berisi semua literal dari dua klausa asli *kecuali* keduanya literal pelengkap. Misalnya, sudah

$$\frac{P_{1,1} \vee P_{3,1}, \quad \neg P_{1,1} \vee \neg P_{2,2}}{P_{3,1} \vee \neg P_{2,2}}.$$

Ada satu lagi aspek teknis dari aturan resolusi: klausa yang dihasilkan harus berisi hanya satu salinan dari setiap literal. [9] Penghapusan beberapa salinan literal disebut **anjak piutang** . Misalnya, jika kita menyelesaikan (A ∨ B) dengan (A ∨ ¬B), kita memperoleh (A ∨ A), yang direduksi menjadi hanya A.

PABRIK

The *kesehatan* dari aturan resolusi dapat dilihat dengan mudah dengan mempertimbangkan literal l_i yang melengkapi untuk literal m_j dalam klausa lain. Jika l_i benar, maka m_j adalah palsu, dan karenanya $m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n$ harus benar, karena $m_1 \vee \cdots \vee m_n$ diberikan. Jika l_i adalah salah, maka $l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k$ harus benar karena $l_1 \vee \cdots \vee l_k$ diberikan. Sekarang Apakah saya benar atau salah, jadi salah satu dari kesimpulan ini berlaku — persis seperti resolusi negara aturan.

Yang lebih mengejutkan tentang aturan resolusi adalah bahwa ia membentuk dasar bagi sebuah keluarga dari *lengkap* prosedur inferensi. *Pepatah teorema berbasis resolusi dapat, untuk setiap kalimat* α *dan* β *dalam logika proposisional, putuskan apakah* $\alpha \models \beta$. Dua subbagian berikutnya menjelaskan bagaimana resolusi mencapai ini.

Bentuk normal konjungtif

Aturan resolusi hanya berlaku untuk klausa (yaitu, disjungsi literal), sehingga tampaknya hanya relevan dengan basis pengetahuan dan pertanyaan yang terdiri dari klausa. Lalu bagaimana bisa itu mengarah pada prosedur inferensi lengkap untuk semua logika proposisional? Jawabannya adalah itu *setiap kalimat dari logika proposisional secara logis setara dengan konjungsi dari klausa.* SEBUAH Kalimat yang diekspresikan sebagai konjungsi dari klausa dikatakan dalam **bentuk normal konjungtif** atau **CNF** (lihat Gambar 7.14). Kami sekarang menjelaskan prosedur untuk mengkonversi ke CNF. Kami menggambarkan prosedur dengan mengubah kalimat $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ menjadi CNF. Langkah-langkahnya adalah sebagai berikut:

KATA PENGHUBUNG
FORMULIR NORMAL

1. Hilangkan ⇔, ganti $\alpha \Leftrightarrow \beta$ dengan $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

   $$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}).$$

2. Hilangkan ⇒, ganti $\alpha \Rightarrow \beta$ dengan $\neg\alpha \vee \beta$:

   $$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg (P_{1,2} \vee P_{2,1}) \vee B_{1,1}).$$

[9] Jika klausa dipandang sebagai *seperangkat* literal, maka batasan ini secara otomatis dihormati. Menggunakan notasi yang diset untuk klausul membuat aturan resolusi jauh lebih bersih, dengan biaya memperkenalkan notasi tambahan.

3. CNF membutuhkan ¬ untuk hanya muncul dalam literal, jadi kami "pindah ¬ ke dalam" oleh aplikasi berulang kation dari persamaan berikut dari Gambar 7.11:

   $\neg(\neg\alpha) \equiv \alpha$ (penghapusan negasi ganda)
   $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ (De Morgan)
   $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ (De Morgan)

   Dalam contoh ini, kami hanya memerlukan satu aplikasi dari aturan terakhir:

   $$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1}).$$

4. Sekarang kita memiliki kalimat yang berisi operator bersarang ∧ dan ∨ diterapkan pada literal. Kita menerapkan hukum distributivitas dari Gambar 7.11, mendistribusikan ∨ lebih ∧ sedapat mungkin.

$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$.

Kalimat aslinya sekarang dalam CNF, sebagai gabungan dari tiga klausa. Jauh lebih sulit baca, tetapi dapat digunakan sebagai input untuk prosedur resolusi.

Algoritma resolusi

Prosedur inferensi berdasarkan resolusi bekerja dengan menggunakan prinsip pembuktian oleh contradiction diperkenalkan pada halaman 250. Artinya, untuk menunjukkan bahwa KB | = α, kami menunjukkan bahwa (KB ∧ ¬α) adalah tidak memuaskan. Kami melakukan ini dengan membuktikan kontradiksi.

Algoritma resolusi ditunjukkan pada Gambar 7.12. Pertama, (KB ∧ ¬α) dikonversi menjadi CNF. Kemudian, aturan resolusi diterapkan pada klausa yang dihasilkan. Setiap pasangan yang berisi literal komplementer diselesaikan untuk menghasilkan klausa baru, yang ditambahkan ke set jika itu belum ada. Proses berlanjut hingga satu dari dua hal terjadi:

- tidak ada klausa baru yang dapat ditambahkan, dalam hal ini KB tidak memerlukan α; atau,
- dua klausa memutuskan untuk menghasilkan klausa *kosong* , dalam hal ini KB mencakup α.

Klausa kosong — disjungsi tanpa disjungsi — sama dengan False karena disjungsi benar hanya jika setidaknya salah satu dari disjunctsnya benar. Cara lain untuk melihat bahwa klausa kosong merupakan suatu kontradiksi untuk mengamati bahwa itu muncul hanya dari menyelesaikan dua komplementer klausa unit seperti P dan ¬P.

Kita dapat menerapkan prosedur resolusi ke kesimpulan yang sangat sederhana di dunia wumpus. Ketika agen berada di [1,1], tidak ada angin, sehingga tidak ada lubang di kotak tetangga. Basis pengetahuan yang relevan adalah

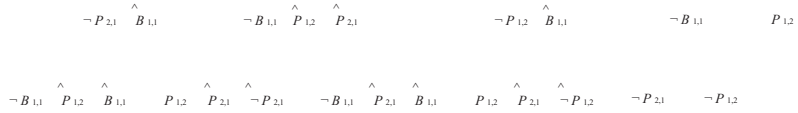
$$KB = R_2 \wedge R_4 = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$$

dan kami ingin membuktikan α yang, katakanlah, $\neg P_{1,2}$. Saat kami mengonversi (KB ∧ ¬α) ke CNF, kami dapatkan klausa yang ditunjukkan di bagian atas Gambar 7.13. Baris kedua dari gambar menunjukkan klausa yang diperoleh dengan menyelesaikan pasangan di baris pertama. Kemudian, ketika $P_{1,2}$ diselesaikan dengan $\neg P_{1,2}$, kita mendapatkan klausa kosong, ditampilkan sebagai kotak kecil. Inspeksi Gambar 7.13 mengungkapkan itu banyak langkah resolusi tidak ada gunanya. Misalnya, klausa $B_{1,1} \vee \neg B_{1,1} \vee P_{1,2}$ setara ke True ∨ $P_{1,2}$ yang setara dengan True. Menyerahkan bahwa Benar itu benar tidak terlalu membantu. Oleh karena itu, setiap klausa di mana dua literal pelengkap muncul dapat diabaikan.

fungsi PL-R ESOLUTION (KB, α) **mengembalikan** benar atau salah
  input : KB, basis pengetahuan, kalimat dalam logika proposisional
    α, kueri, kalimat dalam logika proposisional

  klausa ← himpunan klausa dalam representasi CNF dari KB ∧ ¬α
  baru} {}
  loop lakukan
    untuk setiap pasangan klausa C_i , C_j **dalam** klausa **lakukan**
      penyelesaian ents PL-R ESOLVE (C_i , C_j)
      jika resolvents berisi klausa kosong **maka return** true
      resolusi baru ← baru ∪
    jika ⊆ klausa baru **maka kembalikan** salah
    klausa ← klausa ∪ baru

Gambar 7.12 Algoritma resolusi sederhana untuk logika proposisional. Fungsinya PL-R ESOLVE mengembalikan set semua klausa yang mungkin diperoleh dengan menyelesaikan dua inputnya.



Gambar 7.13 Aplikasi parsial PL-R ESOLUTION untuk inferensi sederhana dalam wumpus dunia. $\neg P_{1,2}$ ditunjukkan mengikuti dari empat klausa pertama di baris atas.

Kelengkapan resolusi

Untuk menyimpulkan diskusi kami tentang resolusi, kami sekarang menunjukkan mengapa PL-R ESOLUTION selesai.

RESOLUSI PENUTUPAN

Untuk melakukan ini, kami memperkenalkan RC **penutupan resolusi** (S) dari satu set klausa S, yang merupakan set dari semua klausa yang dapat diturunkan dengan penerapan berulang dari aturan resolusi untuk klausa dalam S atau mereka turunannya. Penutupan resolusi adalah apa yang dihitung oleh PL-R ESOLUTION sebagai nilai akhir klausa variabel. Sangat mudah untuk melihat bahwa RC (S) harus terbatas, karena hanya ada yang terbatas banyak klausa berbeda yang dapat dibangun dari simbol $P_1, ..., P_k$ yang muncul dalam S. (Perhatikan bahwa ini tidak akan benar tanpa langkah anjak yang menghapus banyak salinan literal.) Karenanya, PL-R ESOLUTION selalu berakhir.

TANAH RESOLUSI DALIL

Teorema kelengkapan untuk resolusi dalam logika proposisional disebut **tanah teorema resolusi** :

> Jika satu set klausa tidak memuaskan, maka resolusi penutupan klausa tersebut berisi klausa kosong.

Teorema ini dibuktikan dengan menunjukkan kontrapositifnya: jika penutupan RC (S) *tidak*

Halaman 23

mengandung klausa kosong, maka S memuaskan. Bahkan, kita bisa membuat model untuk S dengan nilai kebenaran yang sesuai untuk $P_1, ..., P_k$. Prosedur konstruksi adalah sebagai berikut:

> Untuk saya dari 1 hingga k,
>
> - Jika klausa dalam RC (S) berisi $\neg P_i$ literal dan semua literal lainnya salah di bawah penugasan yang dipilih untuk $P_1, ..., P_{i-1}$, lalu tetapkan false ke P_i.
> - Kalau tidak, tetapkan true ke P_i.

Tugas ini untuk $P_1, ..., P_k$ adalah model S. Untuk melihat ini, asumsikan sebaliknya — itu, pada beberapa tahap i dalam urutan, menetapkan simbol P_i menyebabkan beberapa klausa C menjadi salah. Agar hal ini terjadi, pastilah semua literal *lain* dalam C pasti sudah dipalsukan oleh penugasan ke $P_1, ..., P_{i-1}$. Dengan demikian, C sekarang harus terlihat seperti salah (false $\vee$ false $\vee \cdots$ false$\vee P_i$) atau like (false$\vee$false$\vee \cdots \vee$false$\vee\neg P_i$). Jika salah satu dari keduanya ada di RC (S), maka algoritma akan menetapkan nilai kebenaran yang sesuai untuk P_i untuk membuat C benar, jadi C hanya bisa dipalsukan jika *kedua* klausa ini dalam RC (S). Sekarang, karena RC (S) ditutup di bawah resolusi, itu akan mengandung ketetapan dari dua klausa ini, dan ketetapan itu akan memiliki semua literalnya sudah dipalsukan oleh penugasan ke $P_1, ..., P_{i-1}$. Ini bertentangan dengan asumsi kami bahwa klausa palsu pertama muncul pada tahap i. Oleh karena itu, kami telah membuktikan bahwa konstruksinya tidak pernah memalsukan klausa dalam RC (S); yaitu, menghasilkan model RC (S) dan dengan demikian model S itu sendiri (karena S terkandung dalam RC (S)).

7.5.3 Klausa tanduk dan klausa yang pasti

Kelengkapan resolusi menjadikannya metode inferensi yang sangat penting. Secara praktis banyak situasi, bagaimanapun, kekuatan penuh resolusi tidak diperlukan. Beberapa pengetahuan dunia nyata pangkalan memenuhi batasan tertentu pada bentuk kalimat yang dikandungnya, yang memungkinkan mereka untuk menggunakan algoritma inferensi yang lebih terbatas dan efisien.

KETENTUAN DEFINITE

Salah satu bentuk terbatas tersebut adalah **klausa pasti** , yang merupakan disjungsi dari literal yang mana yang *positif* . Misalnya, klausa ($\neg L_{1,1} \vee \neg$Beku $\vee B_{1,1}$) adalah pasti klausa, sedangkan ($\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$) tidak.

KLAUS KUDA

Sedikit lebih umum adalah **klausa Horn** , yang merupakan disjungsi dari literal yang *di kebanyakan orang positif* . Jadi semua klausa yang pasti adalah klausa Horn, seperti klausa tanpa positif literal; ini disebut **klausa tujuan** . Klausa klakson ditutup berdasarkan resolusi: jika Anda menyelesaikannya dua klausa Horn, Anda mendapatkan klausa Horn.

KLAUS TUJUAN

Basis pengetahuan yang hanya berisi klausa tertentu menarik karena tiga alasan:

1. Setiap klausa pasti dapat ditulis sebagai implikasi yang premisnya adalah konjungsi literal positif dan yang kesimpulannya adalah literal positif tunggal. (Lihat Latihan 7.13.) Sebagai contoh, klausa pasti ($\neg L_{1,1} \vee \neg Beku \vee B_{1,1}$) dapat ditulis sebagai implication ($L_{1,1} \wedge Breeze$) $\Rightarrow B_{1,1}$. Dalam bentuk implikasinya, kalimat itu lebih mudah mengerti: dikatakan bahwa jika agen berada di [1,1] dan ada angin, maka [1,1] berangin.

TUBUH
KEPALA
FAKTA

Dalam bentuk Tanduk, premis disebut **tubuh** dan kesimpulannya disebut **kepala**. SEBUAH Kalimat yang terdiri dari satu literal positif tunggal, seperti $L_{1,1}$, disebut **fakta**. Itu juga bisa dituliskan dalam bentuk implikasi sebagai Benar $\Rightarrow L_{1,1}$, tetapi lebih mudah untuk menulis hanya $L_{1,1}$.

Halaman 24

$$\begin{aligned}
\text{CNFSentence} &\rightarrow \text{Klausa}_1 \wedge \cdots \wedge \text{Klausul}_n \\
\text{Klausa} &\rightarrow \text{Literal}_1 \vee \cdots \vee \text{Literal}_m \\
\text{Sastra} &\rightarrow \text{Simbol} \mid \neg\text{Simbol} \\
\text{Simbol} &\rightarrow P \mid Q \mid R \mid \ldots \\
\text{HornClauseForm} &\rightarrow \text{DefiniteClauseForm} \mid \text{GoalClauseForm} \\
\text{DefiniteClauseForm} &\rightarrow (\ \text{Simbol}_1 \wedge \cdots \wedge \text{Simbol}_l\ ) \Rightarrow \text{Simbol} \\
\text{GoalClauseForm} &\rightarrow (\ \text{Simbol}_1 \wedge \cdots \wedge \text{Simbol}_l\ ) \Rightarrow \text{Salah}
\end{aligned}$$

Gambar 7.14 Tata bahasa untuk bentuk normal konjungtif, klausa Horn, dan klausa pasti. Klausa seperti $A \wedge B \Rightarrow C$ masih merupakan klausa yang pasti ketika ditulis sebagai $\neg A \vee \neg B \vee C$, tetapi hanya yang pertama dianggap sebagai bentuk kanonik untuk klausa yang pasti. Satu kelas lagi adalah yang *k* -CNF kalimat, yang merupakan kalimat CNF dimana setiap klausul memiliki paling *k* literal.

RANTAI-MAJU
KE BELAKANG-RANTAI

2. Inferensi dengan klausa Horn dapat dilakukan melalui **forward-chaining** dan **backward-**algoritma **chaining**, yang akan kami jelaskan selanjutnya. Kedua algoritma ini alami, karena langkah-langkah inferensi jelas dan mudah untuk diikuti manusia. Jenis ini inferensi adalah dasar untuk **pemrograman logika**, yang dibahas pada Bab 9.
3. Memutuskan entailment dengan klausa Horn dapat dilakukan dalam waktu yang *linear* dalam ukuran basis pengetahuan — kejutan yang menyenangkan.

7.5.4 Forward forward dan backward

Algoritma forward-chaining PL-FC-E NTAIL ? (KB, q) menentukan apakah suatu proposisi tunggal simbol tion q — permintaan — disyaratkan oleh basis pengetahuan klausa tertentu. Ini berawal dari fakta yang diketahui (literal positif) dalam basis pengetahuan. Jika semua tempat dari suatu Jika diketahui, maka kesimpulannya ditambahkan ke set fakta yang diketahui. Misalnya, jika $L_{1,1}$ dan Breeze diketahui dan ($L_{1,1} \wedge Breeze$) $\Rightarrow B_{1,1}$ ada di basis pengetahuan, maka $B_{1,1}$ bisa ditambahkan. Proses ini berlanjut sampai kueri q ditambahkan atau sampai tidak ada kesimpulan lebih lanjut dibuat. Algoritma terperinci ditunjukkan pada Gambar 7.15; Poin utama yang perlu diingat adalah itu ini berjalan dalam waktu linier.

Cara terbaik untuk memahami algoritma adalah melalui contoh dan gambar. Arure 7.16 (a) menunjukkan basis pengetahuan sederhana klausa Horn dengan A dan B sebagai fakta yang diketahui. Gambar 7.16 (b) menunjukkan basis pengetahuan yang sama dengan **grafik AND – OR** (lihat Babter 4). Dalam grafik AND – ATAU, beberapa tautan yang bergabung dengan sebuah busur menunjukkan suatu konjungsi — setiap tautan harus dibuktikan — sementara banyak tautan tanpa busur menunjukkan disjungsi — tautan apa pun bisa dibuktikan. Sangat mudah untuk melihat bagaimana forward chaining bekerja dalam grafik. Daunnya dikenal (di sini, A dan B) diatur, dan inferensi merambat ke grafik sejauh mungkin. Di manapernah konjungsi muncul, propagasi menunggu sampai semua konjungsi diketahui sebelumnya melanjutkan. Pembaca didorong untuk mengerjakan contoh secara rinci.

Halaman 25


```
fungsi PL-FC-E NTAILS ? (KB, q) mengembalikan benar atau salah
  input : KB, basis pengetahuan, satu set klausa pasti proposisional
          q, kueri, simbol proposisi
  hitung ← tabel, di mana hitung [c] adalah jumlah simbol dalam premis c
  disimpulkan table sebuah tabel, di mana [s] yang disimpulkan awalnya salah untuk semua simbol
  agenda ← antrian simbol, awalnya simbol yang diketahui benar di KB

  sedangkan agenda tidak kosong do
      p ← P OP (agenda)
      jika p = q maka kembalikan benar
      jika disimpulkan [p] = salah maka
          disimpulkan [p] ← benar
          untuk setiap klausa c dalam KB di mana p dalam cP REMISE lakukan
              jumlah pengurangan [c]
              jika menghitung [c] = 0 kemudian menambahkan cC ONCLUSION ke agenda
  kembali salah
```

Gambar 7.15 Algoritma forward-chaining untuk logika proposisional. Agenda terus melacak simbol yang dikenal benar tetapi belum "diproses." Tabel hitungan melacak berapa banyak premis dari setiap implikasi yang belum diketahui. Setiap kali simbol baru p dari agenda diproses, jumlah dikurangi satu untuk setiap implikasi di premis siapa p muncul (mudah diidentifikasi dalam waktu konstan dengan pengindeksan yang sesuai.) Jika suatu hitungan mencapai nol, semua premis implikasi diketahui, sehingga kesimpulannya dapat ditambahkan ke Jadwal acara. Akhirnya, kita perlu melacak simbol mana yang telah diproses; simbol itu sudah di set simbol yang disimpulkan tidak perlu ditambahkan ke agenda lagi. Ini menghindari pekerjaan yang berlebihan dan mencegah loop yang disebabkan oleh implikasi seperti $P \Rightarrow Q$ dan $Q \Rightarrow P$.

Sangat mudah untuk melihat bahwa forward chaining adalah **suara** : setiap kesimpulan pada dasarnya adalah sebuah aplikasi kation dari Modus Ponens. Forward chaining juga **lengkap** : setiap kalimat atom yang disyaratkan akan diturunkan. Cara termudah untuk melihat ini adalah dengan mempertimbangkan keadaan akhir dari tabel yang disimpulkan

TITIK PASTI (setelah algoritma mencapai **titik tetap di** mana tidak ada inferensi baru dimungkinkan). Meja mengandung true untuk setiap simbol yang disimpulkan selama proses, dan false untuk semua simbol lainnya. Kita bisa melihat tabel sebagai model logis; Selain itu, *setiap klausa yang pasti dalam KB asli adalah benar dalam model ini.* Untuk melihat ini, anggap sebaliknya, yaitu bahwa beberapa klausa $a_1 \wedge ... \wedge a_k \Rightarrow b$ salah dalam model. Maka $a_1 \wedge ... \wedge a_k$ harus benar dalam model dan b harus salah dalam model. Tetapi ini bertentangan dengan asumsi kami bahwa algoritma telah mencapai titik tetap! Kita dapat menyimpulkan, oleh karena itu, bahwa set kalimat atom disimpulkan pada titik tetap mendefinisikan model KB asli. Selanjutnya, setiap kalimat atom q yang disyaratkan oleh KB harus benar dalam semua modelnya dan dalam model ini khususnya. Karenanya, setiap atom diperlukan kalimat q harus disimpulkan oleh algoritma.

DATA-DICURI Forward chaining adalah contoh konsep umum penalaran **berbasis data** — itu adalah, alasan di mana fokus perhatian dimulai dengan data yang diketahui. Itu bisa digunakan di dalam agen untuk mendapatkan kesimpulan dari persepsi yang masuk, sering kali tanpa permintaan khusus di pikiran. Sebagai contoh, agen wumpus mungkin T ELL persepsinya ke basis pengetahuan menggunakan

Halaman 26

Gambar 7.16 (A) Satu set klausa Horn. (B) Grafik AND - OR yang sesuai .

algoritme rantai-maju bertahap di mana fakta-fakta baru dapat ditambahkan ke agenda memulai inferensi baru. Pada manusia, sejumlah alasan yang didorong oleh data muncul sebagai hal baru informasi tiba. Misalnya, jika saya di dalam ruangan dan mendengar hujan mulai turun, itu mungkin terjadi bagi saya bahwa piknik akan dibatalkan. Namun mungkin tidak akan terpikir oleh saya bahwa ketujuh belas kelopak bunga mawar terbesar di kebun tetangga saya akan basah; manusia terus maju chaining di bawah kontrol yang hati-hati, jangan sampai mereka dibanjiri dengan konsekuensi yang tidak relevan.

Algoritma backward-chaining, seperti namanya, bekerja mundur dari pertanyaan. Jika kueri q diketahui benar, maka tidak ada pekerjaan yang diperlukan. Kalau tidak, algoritma menemukan implikasi tersebut dalam basis pengetahuan yang kesimpulannya q. Jika semua tempat salah satu implikasi tersebut dapat dibuktikan benar (dengan rantai belakang), maka q adalah benar. Kapan diterapkan pada kueri Q pada Gambar 7.16, ia bekerja kembali grafik sampai mencapai satu set fakta yang diketahui, A dan B, yang membentuk dasar untuk pembuktian. Algoritma pada dasarnya identik untuk algoritma A ND -O R -G RAPH -S EARCH pada Gambar 4.11. Seperti halnya forward chaining, sebuah implementasi yang efisien berjalan dalam waktu linier.

TUJUAN LANGSUNG PEMIKIRAN

Backward chaining adalah bentuk **penalaran yang diarahkan** pada **tujuan** . Ini berguna untuk menjawab pertanyaan spesifik seperti "Apa yang harus saya lakukan sekarang?" dan "Di mana kunci saya?" Seringkali, biayanya rantai mundur *jauh lebih kecil* dari linier dalam ukuran basis pengetahuan, karena proses hanya menyentuh fakta yang relevan.

7,6 E P ERSIAPAN P EROPA M ODEL C HECKING

Pada bagian ini, kami menjelaskan dua keluarga algoritma efisien untuk proposisional umum inferensi berdasarkan pengecekan model: Satu pendekatan berdasarkan pencarian backtracking, dan satu pada pencarian mendaki bukit lokal. Algoritma ini adalah bagian dari "teknologi" proposisional logika. Bagian ini dapat di-skim pada bacaan pertama bab ini.

Halaman 27

Algoritma yang kami jelaskan adalah untuk memeriksa kepuasan: masalah SAT. (Seperti dicatat sebelumnya, pengujian entailment, α | = β, bisa dilakukan dengan menguji *un* satisfiability dari α ∧ ¬β) Kami. telah mencatat hubungan antara menemukan model yang memuaskan untuk kalimat yang logis dan menemukan solusi untuk masalah kepuasan kendala, jadi mungkin tidak mengejutkan kedua keluarga algoritma sangat mirip dengan algoritma backtracking pada Bagian 6.3 dan algoritma pencarian lokal pada Bagian 6.4. Mereka, bagaimanapun, sangat penting dalam hak mereka sendiri karena begitu banyak masalah kombinatorial dalam ilmu komputer dapat dikurangi untuk memeriksa kepuasan kalimat proposisional. Setiap peningkatan kepuasan Algoritma memiliki konsekuensi besar bagi kemampuan kita menangani kompleksitas secara umum.

7.6.1 Algoritma backtracking lengkap

DAVIS – PUTNAM ALGORITMA

Algoritma pertama yang kami pertimbangkan sering disebut **algoritma Davis-Putnam** , setelah sem

makalah akhir oleh Martin Davis dan Hilary Putnam (1960). Algoritma sebenarnya adalah versi dijelaskan oleh Davis, Logemann, dan Loveland (1962), jadi kami akan menyebutnya DPLL setelah awal-semua empat penulis. DPLL mengambil sebagai input kalimat dalam bentuk normal konjungtif — satu set klausa. Seperti B ACKTRACKING -S EARCH dan TT-E NTAILS ?, pada dasarnya adalah sebuah rekursif, enumerasi mendalam-pertama dari model yang mungkin. Ini mewujudkan tiga peningkatan dari yang sederhana skema TT-E NTAILS ?:

- *Pengakhiran dini* : Algoritma mendeteksi apakah kalimat itu harus benar atau salah, bahkan dengan model yang sebagian selesai. Klausa itu benar jika *ada* literal yang benar, bahkan jika literal lainnya belum memiliki nilai kebenaran; karenanya, kalimat secara keseluruhan bisa dinilai benar bahkan sebelum model selesai. Misalnya, kalimat $(A \vee B) \wedge (A \vee C)$ benar jika A benar, terlepas dari nilai B dan C. Demikian pula, kalimat salah jika *ada* klausa yang salah, yang terjadi ketika masing-masing literalnya salah. Lagi-lagi ini dapat terjadi jauh sebelum model selesai. Pengakhiran dini menghindari pemeriksaan seluruh sub pohon di ruang pencarian.

SIMBOL MURNI

- *Pure simbol heuristik* : Sebuah **simbol murni** adalah simbol yang selalu muncul dengan sama "Masuk" di semua klausa. Misalnya, dalam tiga klausa $(A \vee \neg B)$, $(\neg B \vee \neg C)$, dan $(C \vee A)$, simbol A murni karena hanya literal positif yang muncul, B murni karena hanya literal negatif yang muncul, dan C tidak murni. Sangat mudah untuk melihat bahwa jika sebuah kalimat memiliki model, kemudian memiliki model dengan simbol-simbol murni yang ditugaskan untuk membuat literal mereka benar, karena hal itu tidak pernah bisa membuat klausa salah. Perhatikan bahwa dalam menentukan kemurnian simbol, algoritma dapat mengabaikan klausa yang sudah ada diketahui benar dalam model yang dibangun sejauh ini. Misalnya, jika modelnya berisi B = false, maka klausa $(\neg B \vee \neg C)$ sudah benar, dan dalam klausa yang tersisa C hanya muncul sebagai literal positif; karena itu C menjadi murni.
- *heuristik Satuan klausul* : Sebuah **klausul Unit** didefinisikan sebelumnya sebagai klausul dengan hanya satu literal. Dalam konteks DPLL, itu juga berarti klausa di mana semua literal kecuali satu sudah ditetapkan salah oleh model. Misalnya, jika model mengandung B = true, kemudian $(\neg B \vee \neg C)$ menyederhanakan menjadi $\neg C$, yang merupakan klausa satuan. Jelas, untuk klausa ini untuk menjadi benar, C harus disetel ke false. Heuristik klausa unit memberikan semua simbol tersebut sebelum bercabang pada sisanya. Salah satu konsekuensi penting dari heuristik adalah itu

Halaman 28


```
function DPLL-SATISFIABLE? (s) mengembalikan benar atau salah
  input : s, kalimat dalam logika proposisional

  klausa ← himpunan klausa dalam representasi CNF dari s
  simbol ← daftar simbol proposisi di s
  mengembalikan DPLL (klausa, simbol, {})

fungsi DPLL (klausa, simbol, model) mengembalikan benar atau salah

  jika setiap klausa dalam klausa benar dalam model maka kembalikan benar
  jika beberapa klausa dalam klausa salah dalam model maka kembalikan salah
  P, nilai ← FIND-PURE-SYMBOL (simbol, klausa, model)
  jika P bukan nol, maka kembalikan DPLL (klausa, simbol - P, model ∪ {P = nilai})
  P, nilai ← FIND-UNIT-CLAUSE (klausa, model)
  jika P bukan nol, maka kembalikan DPLL (klausa, simbol - P, model ∪ {P = nilai})
  P ← FIRST (simbol); sisanya ← REST (simbol)
  mengembalikan DPLL (klausa, istirahat, model ∪ {P = true}) atau
          DPLL (klausa, istirahat, model ∪ {P = false}))
```

Gambar 7.17 Algoritma DPLL untuk memeriksa kepuasan suatu kalimat secara proposisional logika. Gagasan di balik FIND-PURE-SYMBOL dan FIND-UNIT-CLAUSE dijelaskan dalam teks; masing-masing mengembalikan simbol (atau nol) dan nilai kebenaran untuk ditetapkan ke simbol itu. Suka TT-ENTAILS ?, DPLL beroperasi pada model parsial.

setiap upaya untuk membuktikan (dengan sanggahan) literal yang sudah ada dalam basis pengetahuan akan segera berhasil (Latihan 7.23). Perhatikan juga bahwa menetapkan satu unit klausa bisa

buat klausa unit lain — misalnya, ketika C disetel ke false, (C ∨ A) menjadi a unit klausa, menyebabkan true ditugaskan ke A. "kaskade" ini dari tugas paksa

PROPAGASI UNIT

disebut **unit propagasi** . Itu menyerupai proses forward chaining dengan pasti klausa, dan memang, jika ekspresi CNF hanya berisi klausa pasti maka DPLL pada dasarnya replikasi forward chaining. (Lihat Latihan 7.24.)

Algoritma DPLL ditunjukkan pada Gambar 7.17, yang memberikan kerangka esensi dari proses pencarian.

Apa yang tidak diperlihatkan oleh Gambar 7.17 adalah trik yang memungkinkan pemecah SAT meningkat masalah besar. Sangat menarik bahwa sebagian besar trik ini sebenarnya agak umum, dan kami pernah melihat mereka sebelumnya dengan samaran lain:

1. **Analisis komponen** (seperti yang terlihat dengan Tasmania dalam CSP): Seperti DPLL memberikan nilai-nilai kebenaran untuk variabel, set klausa dapat dipisahkan menjadi himpunan bagian terpisah, yang disebut **components** , yang tidak berbagi variabel yang tidak ditetapkan. Diberikan cara efisien untuk mendeteksi kapan ini Jika terjadi, pemecah dapat memperoleh kecepatan yang cukup dengan mengerjakan setiap komponen secara terpisah.
2. **Variabel dan pemesanan nilai** (seperti yang terlihat di Bagian 6.3.1 untuk CSP): Penerapan sederhana kami mentasi DPLL menggunakan pemesanan variabel arbitrer dan selalu mencoba nilai yang *benar* sebelum *salah* . The **gelar heuristik** (lihat halaman 216) menyarankan memilih variabel yang muncul paling sering di atas semua klausa yang tersisa.

Halaman 29

3. **Pelacakan yang cerdas** (seperti yang terlihat di Bagian 6.3 untuk CSP): Banyak masalah yang dapat tidak dapat diselesaikan dalam jam waktu berjalan dengan pengulangan kronologis dapat diselesaikan di detik dengan backtracking cerdas yang mencadangkan semua jalan ke titik yang relevan konflik. Semua pemecah SAT yang melakukan backtracking cerdas menggunakan beberapa bentuk **konflik klausa belajar** merekam konflik sehingga tidak akan terulang lagi dalam pencarian. Biasanya satu set konflik ukuran terbatas disimpan, dan yang jarang digunakan dijatuhkan.
4. **Restart acak** (seperti yang terlihat pada halaman 124 untuk mendaki bukit): Kadang-kadang lari tampaknya tidak untuk membuat kemajuan. Dalam hal ini, kita dapat memulai dari atas pohon pencarian, daripada mencoba melanjutkan. Setelah restart, pilihan acak berbeda (dalam variabel dan pemilihan nilai) dibuat. Klausa yang dipelajari dalam menjalankan pertama dipertahankan setelah restart dan dapat membantu memangkas ruang pencarian. Restart tidak menjamin bahwa a solusi akan ditemukan lebih cepat, tetapi mengurangi varians pada waktu untuk solusi.
5. **Pengindeksan pintar** (seperti yang terlihat dalam banyak algoritma): Metode percepatan yang digunakan dalam DPLL itu sendiri, serta trik yang digunakan dalam pemecah modern, memerlukan pengindeksan cepat dari hal-hal seperti itu sebagai "himpunan klausa di mana variabel X_i muncul sebagai literal positif." Tugas ini adalah diperumit oleh fakta bahwa algoritma hanya tertarik pada klausa yang dimiliki belum puas dengan tugas sebelumnya untuk variabel, jadi struktur pengindeksan harus diperbarui secara dinamis saat proses perhitungan berlangsung.

Dengan peningkatan ini, pemecah modern dapat menangani masalah dengan puluhan juta variasi ables Mereka telah merevolusi bidang-bidang seperti verifikasi perangkat keras dan protokol keamanan verifikasi, yang sebelumnya membutuhkan bukti dipandu tangan yang melelahkan.

7.6.2 Algoritma pencarian lokal

Kami telah melihat beberapa algoritma pencarian lokal sejauh ini dalam buku ini, termasuk H ILL -C LIMBING (halaman 122) dan S IMULATED -A NNEALING (halaman 126). Algoritma ini dapat diterapkan di- jelas untuk masalah kepuasan, asalkan kita memilih fungsi evaluasi yang tepat. Menjadi- sebab tujuannya adalah untuk menemukan tugas yang memenuhi setiap klausa, fungsi evaluasi itu menghitung jumlah klausa yang tidak puas akan melakukan pekerjaan. Sebenarnya, ini adalah ukuran yang tepat digunakan oleh algoritma ONFLICTS M IN -C untuk CSP (halaman 221). Semua algoritma ini mengambil langkah di ruang penugasan lengkap, membalik nilai kebenaran dari satu simbol pada suatu waktu. Itu ruang biasanya mengandung banyak minimum lokal, untuk melarikan diri dari berbagai bentuk acak Dibutuhkan. Dalam beberapa tahun terakhir, ada banyak percobaan untuk menemukan keseimbangan yang baik antara keserakahan dan keacakan.

Salah satu algoritma paling sederhana dan paling efektif untuk muncul dari semua pekerjaan ini disebut W ALK SAT (Gambar 7.18). Pada setiap iterasi, algoritma memilih klausa yang tidak puas dan

mengambil simbol dalam klausa untuk membalik. Ini memilih secara acak antara dua cara untuk memilih yang mana simbol untuk membalik: (1) langkah "konflik mini" yang meminimalkan jumlah klausa yang tidak puas di negara baru dan (2) langkah "jalan acak" yang mengambil simbol secara acak.

Ketika W ALK SAT mengembalikan model, kalimat input memang memuaskan, tetapi ketika itu mengembalikan kegagalan, ada dua kemungkinan penyebab: baik kalimatnya tidak memuaskan atau kita perlu memberikan algoritma lebih banyak waktu. Jika kita mengatur max flips = ∞ dan p> 0, W ALK SAT akan akhirnya mengembalikan model (jika ada), karena langkah berjalan acak akhirnya akan mengenai

Halaman 30


```
fungsi W ALK SAT (klausa, p, maks membalik) mengembalikan model yang memuaskan atau kegagalan
  input : klausa, seperangkat klausa dalam logika proposisional
          p, probabilitas memilih untuk melakukan gerakan "jalan acak", biasanya sekitar 0,5
          max flips, jumlah flips yang diperbolehkan sebelum menyerah

  model ← penugasan acak benar / salah pada simbol dalam klausa
  untuk i = 1 hingga maks membalik lakukan
      jika model memenuhi klausa maka kembalikan model
      klausa ← klausa yang dipilih secara acak dari klausa yang salah dalam model
      dengan probabilitas p membalikkan nilai dalam model simbol yang dipilih secara acak dari klausa
      lain membalik simbol mana pun dalam klausa memaksimalkan jumlah klausa yang puas
  kegagalan pengembalian
```

Gambar 7.18 Algoritma W ALK SAT untuk memeriksa kepuasan dengan membalik secara acak nilai-nilai variabel. Banyak versi algoritme yang ada.

atas solusinya. Sayangnya, jika maks membalik adalah tak terhingga dan kalimat tidak memuaskan, maka Algoritma tidak pernah berakhir!

Untuk alasan ini, W ALK SAT paling berguna ketika kita mengharapkan solusi ada — misalnya cukup, masalah yang dibahas dalam Bab 3 dan 6 biasanya memiliki solusi. Di samping itu, W ALK SAT tidak selalu dapat mendeteksi *ketidakpuasan* , yang diperlukan untuk memutuskan pengalihan. Misalnya, agen tidak dapat *diandalkan* menggunakan W ALK SAT untuk membuktikan bahwa kuadrat aman di dunia wumpus. Sebaliknya, itu bisa mengatakan, “Saya memikirkannya selama satu jam dan tidak bisa memunculkannya dunia yang mungkin di mana alun-alun *tidak* aman. "Ini mungkin merupakan indikator empiris yang baik itu alun-alun itu aman, tapi itu jelas bukan bukti.

7.6.3 Pemandangan masalah SAT acak

Beberapa masalah SAT lebih sulit daripada yang lain. Masalah yang *mudah* bisa diselesaikan dengan algoritma lama rithm, tetapi karena kita tahu bahwa SAT adalah NP-complete, setidaknya beberapa contoh masalah harus membutuhkan run time eksponensial. Dalam Bab 6, kami melihat beberapa penemuan mengejutkan tentang tertentu macam masalah. Sebagai contoh, masalah n-queens — dianggap cukup rumit untuk back-pelacakan algoritma pencarian — ternyata mudah untuk metode pencarian lokal, seperti konflik min. Ini karena solusi terdistribusi sangat padat dalam ruang penugasan KASIH, dan setiap tugas awal dijamin memiliki solusi di dekatnya. Jadi, n-queens adalah mudah karena tidak **dibatasi** .

TIDAK DIKENDALIKAN

Ketika kita melihat masalah kepuasan dalam bentuk normal konjungtif, masalah tegang adalah satu dengan klausa yang relatif *sedikit* membatasi variabel. Sebagai contoh, di sini adalah kalimat 3-CNF yang dibuat secara acak dengan lima simbol dan lima klausa:

$$(\neg D \vee B \neg \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C).$$

Enam belas dari 32 penugasan yang mungkin adalah model dari kalimat ini, jadi, rata-rata, itu akan menjadi ambil hanya dua tebakan acak untuk menemukan model. Ini adalah masalah kepuasan yang mudah, juga

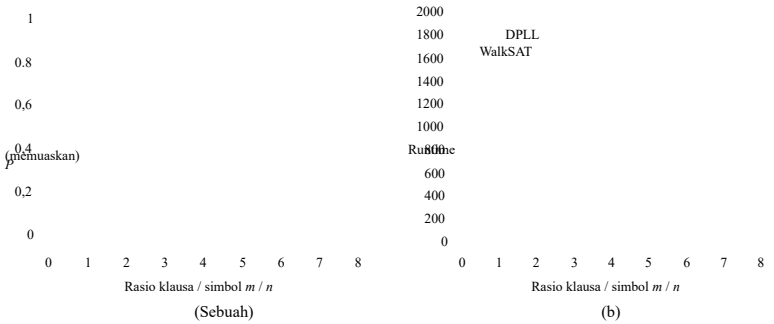
Halaman 31

kebanyakan masalah yang tidak dibatasi seperti itu. Di sisi lain, ada masalah yang terlalu *terbatas* banyak klausa relatif terhadap jumlah variabel dan cenderung tidak memiliki solusi.

Untuk melampaui intuisi dasar ini, kita harus mendefinisikan dengan tepat bagaimana kalimat acak dihasilkan. Notasi CNF $_k$ (m, n) menunjukkan kalimat k-CNF dengan m klausa dan n simbol, di mana klausa dipilih secara seragam, mandiri, dan tanpa penggantian dari antara semua klausa dengan k literal yang berbeda, yang positif atau negatif secara acak. (SEBUAH simbol mungkin tidak muncul dua kali dalam satu klausa, juga tidak mungkin sebuah klausa muncul dua kali dalam sebuah kalimat.)

Diberikan sumber kalimat acak, kita dapat mengukur probabilitas kepuasan. Gambar 7.19 (a) plot probabilitas untuk CNF $_3$ (m, 50), yaitu, kalimat dengan 50 variabel dan 3 liter per klausa, sebagai fungsi dari rasio / simbol, m / n. Seperti yang kita harapkan, untuk kecil m / n probabilitas kepuasan dekat dengan 1, dan pada besar m / n probabilitas mendekati 0. Probabilitas turun cukup tajam di sekitar m / n = 4.3. Secara empiris, kami temukan bahwa "tebing" berada di tempat yang kira-kira sama (untuk k = 3) dan menjadi lebih tajam dan lebih tajam seperti n meningkat. Secara teoritis, **dugaan ambang batas kepuasan** mengatakan bahwa untuk setiap k ≥ 3, ada rasio ambang batas r_k sedemikian sehingga, saat n menuju tak terhingga, probabilitas bahwa CNF $_k$ (n, rn) memuaskan menjadi 1 untuk semua nilai r di bawah ambang batas, dan 0 untuk semua nilai di atas. Dugaan itu tetap tidak terbukti.

KEPUASAN AMBANG DUGAAN



Gambar 7.19 (a) Grafik yang menunjukkan probabilitas bahwa kalimat 3-CNF acak dengan n = 50 simbol memuaskan, sebagai fungsi dari rasio / simbol m / n. (B) Grafik median run time (diukur dalam jumlah panggilan rekursif ke DPLL, proxy yang bagus) secara acak 3-CNF kalimat. Masalah yang paling sulit memiliki rasio klausa / simbol sekitar 4,3.

Sekarang kami memiliki ide yang baik di mana masalah yang memuaskan dan tidak memuaskan, yaitu pertanyaan selanjutnya adalah, di mana masalah yang sulit? Ternyata mereka juga sering ada di nilai ambang batas. Gambar 7.19 (b) menunjukkan bahwa masalah 50-simbol pada nilai ambang 4,3 sekitar 20 kali lebih sulit untuk dipecahkan dibandingkan dengan rasio 3,3. Yang terbatasi masalah paling mudah dipecahkan (karena sangat mudah untuk menebak solusinya); yang dibatasi masalah tidak semudah yang dibatasi, tetapi masih jauh lebih mudah daripada yang benar di ambang pintu.

Halaman 32

7.7 A GENTS B DITEMUKAN PADA P ROPOSITIONAL L OGIC

Pada bagian ini, kami menyatukan apa yang telah kami pelajari sejauh ini untuk membangun wumpus agen dunia yang menggunakan logika proposisional. Langkah pertama adalah mengaktifkan agen untuk menyimpulkan, ke sedapat mungkin, keadaan dunia mengingat sejarah perseptinya. Ini membutuhkan penulisan a model logis lengkap dari efek tindakan. Kami juga menunjukkan bagaimana agen dapat melacak dunia secara efisien tanpa kembali ke sejarah persepsi untuk setiap kesimpulan. Akhirnya, kami menunjukkan bagaimana agen dapat menggunakan inferensi logis untuk membuat rencana yang dijamin mencapai tujuannya.

7.7.1 Keadaan dunia saat ini

Seperti yang dinyatakan di awal bab, agen logis beroperasi dengan menyimpulkan apa yang harus dilakukan dari basis pengetahuan kalimat tentang dunia. Basis pengetahuan terdiri dari aksioma — pengetahuan umum tentang cara kerja dunia — dan persepsi tentang kalimat yang diperoleh dari pengalaman agen di dunia tertentu. Di bagian ini, kami fokus pada masalah menyimpulkan keadaan dunia wumpus saat ini — di mana aku, apakah kotak itu aman, dan sebagainya.

Kami mulai mengumpulkan aksioma di Bagian 7.4.3. Agen tahu bahwa titik awal tidak mengandung pit ($\neg P_{1,1}$) dan tidak ada wumpus ($\neg W_{1,1}$). Selanjutnya, untuk setiap kotak, ia tahu itu kuadrat itu berangin jika dan hanya jika sebuah kotak tetangga memiliki lubang; dan persegi bau jika dan hanya jika kotak tetangga memiliki wumpus. Dengan demikian, kami menyertakan banyak koleksi kalimat dari bentuk berikut:

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$
$$S_{1,1} \Leftrightarrow (W_{1,2} \vee W_{2,1})$$
$$\cdots$$

Agen itu juga tahu persis ada satu wumpus. Ini dinyatakan dalam dua bagian. Pertama, kita harus mengatakan bahwa *setidaknya ada satu* wumpus:

$$W_{1,1} \vee W_{1,2} \vee \cdots \vee W_{4,3} \vee W_{4,4} .$$

Kemudian, kita harus mengatakan bahwa *paling banyak ada satu* wumpus. Untuk setiap pasangan lokasi, kami menambahkan a kalimat yang mengatakan bahwa setidaknya satu dari mereka harus bebas wumpus:

$$\neg W_{1,1} \vee \neg W_{1,2}$$
$$\neg W_{1,1} \vee \neg W_{1,3}$$
$$\cdots$$
$$\neg W_{4,3} \vee \neg W_{4,4} .$$

Sejauh ini bagus. Sekarang mari kita pertimbangkan persepsi agen. Jika saat ini ada bau busuk, satu mungkin mengira bahwa proposisi Stench harus ditambahkan ke basis pengetahuan. Ini bukan cukup benar, namun: jika tidak ada bau busuk pada langkah waktu sebelumnya, maka ¬Stench akan selalu siap ditegaskan, dan pernyataan baru hanya akan menghasilkan kontradiksi. Masalah terpecahkan ketika kita menyadari bahwa persepsi menyatakan sesuatu *hanya tentang waktu saat ini* . Jadi, jika langkah waktu (seperti yang disediakan untuk MAKE-PERCEPT-SENTENCE pada Gambar 7.1) adalah 4, maka kita tambahkan

Halaman 33

Bau busuk$_4$ untuk basis pengetahuan, daripada bau-rapi menghindari pertentangan dengan ¬Schene$_3$. Hal yang sama berlaku untuk persepsi angin, benjolan, kilau, dan teriakan.

Gagasan mengaitkan proposisi dengan langkah-langkah waktu meluas ke segala aspek dunia yang berubah seiring waktu. Sebagai contoh, basis pengetahuan awal termasuk $L^0_{1,1}$ — agen ada di kuadrat [1,1] pada waktu 0 — serta FacingEast$_0$, HaveArrow$_0$, dan WumpusAlive$_0$. Kita gunakan

FASIH kata **fasih** (dari *fluens* Latin , mengalir) untuk merujuk aspek dunia yang berubah. "Lancar" adalah sinonim untuk "variabel negara," dalam arti yang dijelaskan dalam diskusi tentang faktor representasi dalam Bagian 2.4.7 di halaman 57. Simbol yang terkait dengan aspek permanen

ATEMPORAL VARIABEL dunia tidak membutuhkan catatan waktu dan kadang-kadang disebut **variabel temporal** .

Kita dapat menghubungkan bau busuk dan angin langsung ke properti kotak tempat mereka berpengalaman melalui lokasi yang lancar sebagai berikut.[10] Untuk setiap langkah waktu t dan persegi [x, y], kami tegaskan

$$L^t_{x,y} \Rightarrow (\text{Breeze}_t \Leftrightarrow B_{x,y})$$

$L_{x,y} \Rightarrow (Bau_t \Leftrightarrow S_{x,y})$.

Sekarang, tentu saja, kita perlu aksioma yang memungkinkan agen untuk melacak fluents seperti $L_{x,y}^t$. Fasih ini berubah sebagai hasil dari tindakan yang diambil oleh agen, jadi, dalam terminologi Bab 3, kita perlu menuliskan **model transisi** dunia wumpus sebagai satu set kalimat logis.

Pertama, kita perlu simbol proposisi untuk kemunculan tindakan. Seperti halnya persepsi, simbol-simbol ini diindeks berdasarkan waktu; dengan demikian, $Forward_0$ berarti bahwa agen mengeksekusi *Forward* action at time 0. Dengan konvensi, persepsi untuk langkah waktu tertentu terjadi terlebih dahulu, diikuti oleh tindakan untuk langkah waktu itu, diikuti oleh transisi ke langkah waktu berikutnya.

EFEK AXIOM

Untuk menjelaskan bagaimana dunia berubah, kita dapat mencoba menulis **aksioma efek** yang menentukan hasil dari suatu tindakan pada langkah waktu berikutnya. Misalnya, jika agen berada di lokasi [1,1] menghadap timur pada waktu 0 dan Maju, hasilnya adalah bahwa agen berada di bujur sangkar [2,1] dan tidak lagi ada di [1,1]:

$$L^0_{1,1} \wedge FacingEast_0 \wedge Maju_0 \Rightarrow (L^1_{2,1} \wedge \neg L^1_{1,1}). \quad (7.1)$$

Kami akan membutuhkan satu kalimat seperti itu untuk setiap langkah waktu yang mungkin, untuk masing-masing dari 16 kotak, dan masing-masing dari empat orientasi. Kami juga membutuhkan kalimat serupa untuk tindakan lain: *Grab* , *Shoot* , *Climb* , *TurnLeft* , dan *TurnRight* .

Mari kita anggap bahwa agen memutuskan untuk bergerak *Maju* pada waktu 0 dan menegaskan ini fakta menjadi basis pengetahuannya. Diberikan efek aksioma dalam Persamaan (7.1), dikombinasikan dengan pernyataan awal tentang keadaan pada waktu 0, agen sekarang dapat menyimpulkan bahwa ia berada dalam [2,1]. Bahwa adalah, $A_{SK}(KB, L^1_{2,1}) =$ benar. Sejauh ini bagus. Sayangnya, berita di tempat lain kurang bagus: jika kita $A_{SK}(KB, HaveArrow_1)$, jawabannya salah, artinya, agen masih belum bisa membuktikannya memiliki panah; juga tidak dapat membuktikannya *tidak* memilikinya! Informasi telah hilang karena efek aksioma gagal untuk menyatakan apa yang tetap *tidak berubah* sebagai akibat dari suatu tindakan. Kebutuhan untuk dilakukan ini menimbulkan **masalah frame** . [11] Salah satu solusi yang mungkin untuk masalah bingkai akan

MASALAH BINGKAI

[10] Bagian 7.4.3 dengan mudah mengabaikan persyaratan ini.

[11] Nama "masalah bingkai" berasal dari "kerangka referensi" dalam fisika — latar belakang stasioner yang diasumsikan sehubungan dengan gerakan yang diukur. Ini juga memiliki analogi dengan bingkai film, di mana biasanya sebagian besar latar belakang tetap konstan sementara perubahan terjadi di latar depan.

Halaman 34

BINGKAI AXIOM

adalah dengan menambahkan **aksioma bingkai** secara eksplisit menyatakan semua proposisi yang tetap sama. Untuk contoh, untuk setiap kali t kita akan memiliki

$$Maju_t \Rightarrow (HaveArrow_t \Leftrightarrow HaveArrow_{t+1})$$
$$Maju_t \Rightarrow (WumpusAlive_t \Leftrightarrow WumpusAlive_{t+1})$$
$$\cdots$$

di mana kami secara eksplisit menyebutkan setiap proposisi yang tetap tidak berubah dari waktu t ke waktu t +1 di bawah tindakan Maju. Meskipun agen sekarang tahu bahwa ia masih memiliki panah setelah bergerak maju dan bahwa wumpus belum mati atau hidup kembali, proliferasi aksioma bingkai tampaknya sangat tidak efisien. Di dunia dengan m tindakan yang berbeda dan n fasih, himpunan aksioma bingkai akan berukuran O (mn). Manifestasi spesifik ini masalah bingkai kadang-kadang disebut **masalah bingkai representasional** . Secara historis, masalah adalah masalah yang signifikan bagi para peneliti AI; kami mengeksplorasi lebih lanjut dalam catatan di bagian akhir bab ini.

PERWAKILAN MASALAH BINGKAI

Masalah kerangka representasional adalah signifikan karena dunia nyata memiliki sangat banyak lancar, secara halus. Untungnya bagi kita manusia, setiap tindakan biasanya tidak berubah lagi dari sejumlah kecil k dari fasih itu — dunia menunjukkan **lokalitas** . Memecahkan representasi masalah bingkai sentasional membutuhkan pendefinisian model transisi dengan seperangkat aksioma ukuran O (mk) daripada ukuran O (mn). Ada juga **masalah kerangka inferensial** : masalah memproyeksikan hasil pada langkah rencana tindakan dalam waktu O (kt) daripada O (nt).

LOKALITAS

BINGKAI INFERENSI MASALAH

Solusi untuk masalah ini melibatkan mengubah fokus seseorang dari menulis aksioma tentang *tindakan* untuk menulis aksioma tentang *fasih* . Jadi, untuk setiap F lancar, kita akan memiliki aksioma itu mendefinisikan nilai kebenaran dari F^{t+1} dalam hal fasih (termasuk F itu sendiri) pada waktu t dan tindakan yang mungkin terjadi pada waktu t. Sekarang, nilai kebenaran F^{t+1} dapat diatur dalam satu dari dua cara: baik tindakan pada waktu t menyebabkan F benar pada t + 1, atau F sudah benar pada waktu t dan aksi pada waktu t tidak menyebabkannya salah. Aksioma bentuk ini disebut negara **penerus**

SUCCESSOR-STATE AKSIOMA

aksioma dan memiliki skema ini:

$$F^{t+1} \Leftrightarrow ActionCausesF^{t} \vee (F^{t} \wedge \neg ActionCausesNotF^{t}).$$

Salah satu aksioma negara penerus yang paling sederhana adalah aksi untuk HaveArrow. Karena tidak ada tindakan untuk reload, yang $ActionCausesF^{t}$ bagian hilang dan kita dibiarkan dengan

$$HaveArrow^{t+1} \Leftrightarrow (HaveArrow^{t} \wedge \neg Shoot^{t}). \quad (7.2)$$

Untuk lokasi agen, aksioma negara penerus lebih rumit. Misalnya, $L^{t+1}_{1,1}$ benar jika salah satu (a) agen bergerak *Maju* dari [1,2] ketika menghadap ke selatan, atau dari [2,1] saat menghadap ke barat; atau (b) $L^{t}_{1,1}$ sudah benar dan tindakan tidak menyebabkan gerakan (baik karena aksinya tidak Maju atau karena aksinya menabrak tembok). Ditulis dalam logika proposisional, ini menjadi

$$\begin{aligned} L^{t+1}_{1,1} \Leftrightarrow\ & (L^{t}_{1,1} \wedge (\neg Maju^{t} \vee Bump^{t+1})) \\ & \vee (L^{t}_{1,2} \wedge (South^{t} \wedge Forward^{t})) \\ & \vee (L^{t}_{2,1} \wedge (Barat^{t} \wedge Maju^{t})). \end{aligned} \quad (7.3)$$

Latihan 7.26 meminta Anda untuk menulis aksioma untuk sisa dunia fasih wumpus.

Halaman 35

Diberikan set lengkap aksioma negara penerus dan aksioma lain yang terdaftar di awal- Pada bagian ini, agen akan dapat A SK dan menjawab pertanyaan yang dapat dijawab keadaan dunia saat ini. Misalnya, dalam Bagian 7.2 urutan awal dari persepsi dan tindakan adalah

$\neg Stench^{0} \wedge \neg Breeze^{0} \wedge \neg Glitter^{0} \wedge \neg Bump^{0} \wedge \neg Cream^{0}$; $Maju^{0}$

$\neg Stench^{1} \wedge Breeze^{1} \wedge \neg Glitter^{1} \wedge \neg Bump^{1} \wedge \neg Cream^{1}$; $TurnRight^{1}$

$\neg Stench^{2} \wedge Breeze^{2} \wedge \neg Glitter^{2} \wedge \neg Bump^{2} \wedge \neg Cream^{2}$; $TurnRight^{2}$

$\neg Stench^{3} \wedge Breeze^{3} \wedge \neg Glitter^{3} \wedge \neg Bump^{3} \wedge \neg Cream^{3}$; $Maju^{3}$

$\neg Stench^{4} \wedge \neg Beku^{4} \wedge \neg Glitter^{4} \wedge \neg Bump^{4} \wedge \neg Cream^{4}$; $TurnRight^{4}$

$\neg Stench^{5} \wedge \neg Beku^{5} \wedge \neg Glitter^{5} \wedge \neg Bump^{5} \wedge \neg Cream^{5}$; $Maju^{5}$

$Stench^{6} \wedge \neg Beku^{6} \wedge \neg Glitter^{6} \wedge \neg Bump^{6} \wedge \neg cream^{6}$

Pada titik ini, kami memiliki A SK $(KB, L^{6}_{1,2}) =$ benar, jadi agen tahu di mana itu. Bahkan, A SK $(KB, W_{1,3}) =$ true dan A SK $(KB, P_{3,1}) =$ true, sehingga agen telah menemukan wumpus dan salah satu lubang. Pertanyaan yang paling penting untuk agen adalah apakah kotak itu OK untuk bergerak ke dalam, yaitu, alun-alun tidak mengandung lubang atau wumpus hidup. Lebih mudah untuk menambahkan aksioma ini, memiliki bentuk

$$Ok^{t}_{x,y} \Leftrightarrow \neg P_{x,y} \wedge \neg (W_{x,y} \wedge WumpusAlive^{t}).$$

Akhirnya, A SK $(KB, OK^{6}_{2,2}) =$ true, jadi kuadrat [2,2] tidak masalah untuk pindah ke. Padahal, diberikan a Dengan suara dan algoritma inferensi lengkap seperti DPLL, agen dapat menjawab pertanyaan yang dapat dijawab pertanyaan tentang kotak mana yang OK — dan dapat melakukannya hanya dalam beberapa milidetik untuk skala kecil dunia wumpus sedang.

Memecahkan masalah kerangka representasional dan inferensial adalah langkah besar ke depan, tetapi masalah merusak tetap: kita perlu mengkonfirmasi bahwa *semua* prasyarat yang diperlukan dari sebuah tindakan terus untuk itu memiliki efek yang diinginkan. Kami mengatakan bahwa tindakan Maju menggerakkan agen di depan kecuali ada dinding di jalan, tetapi ada banyak pengecualian tidak biasa lainnya yang bisa menyebabkan tindakan gagal: agen mungkin tersandung dan jatuh, terserang serangan jantung, dilakukan pergi oleh kelelawar raksasa, dll. Menentukan semua pengecualian ini disebut **masalah kualifikasi** .

KUALIFIKASI MASALAH

Tidak ada solusi lengkap dalam logika; perancang sistem harus menggunakan penilaian yang baik dalam memutuskan seberapa rinci mereka ingin menentukan model mereka, dan detail apa yang mereka inginkan untuk pergi. Kita akan melihat di Bab 13 bahwa teori probabilitas memungkinkan kita untuk merangkum semua itu pengecualian tanpa menyebutkan nama mereka secara eksplisit.

7.7.2 Agen hibrida

Kemampuan untuk menyimpulkan berbagai aspek dari keadaan dunia dapat dikombinasikan secara langsung.

AGEN HYBRID

diteruskan dengan aturan kondisi-tindakan dan dengan algoritma pemecahan masalah dari Bab 3 dan 4 untuk menghasilkan **agen hibrida** untuk dunia wumpus. Gambar 7.20 menunjukkan satu cara yang memungkinkan untuk melakukan ini. Program agen memelihara dan memperbarui basis pengetahuan serta arus rencana. Basis pengetahuan awal berisi aksioma *atemporal* — yang tidak bergantung pada t, seperti aksioma yang mengaitkan kenyamanan kotak dengan keberadaan lubang. Setiap langkah waktu, kalimat persepsi baru ditambahkan bersama dengan semua aksioma yang bergantung pada t, seperti

sebagai aksioma negara penerus. (Bagian selanjutnya menjelaskan mengapa agen tidak membutuhkan aksioma untuk langkah waktu di *masa depan* .) Kemudian, agen menggunakan inferensi logis, dengan A_{SK} ing pertanyaan dari basis pengetahuan, untuk mengetahui kotak mana yang aman dan mana yang belum dikunjungi.

Badan utama program agen menyusun rencana berdasarkan prioritas yang menurun tujuan. Pertama, jika ada kilau, program membangun rencana untuk mengambil emas, ikuti rute kembali ke lokasi awal, dan memanjat keluar dari gua. Kalau tidak, jika tidak ada rencana saat ini, program merencanakan rute ke kotak aman terdekat yang belum dikunjungi, memastikan rute hanya melewati kotak aman. Perencanaan rute dilakukan dengan A^* cari, bukan dengan A_{SK} . Jika tidak ada kotak aman untuk dijelajahi, langkah selanjutnya — jika agen masih memiliki panah — adalah untuk mencoba membuat kotak yang aman dengan menembak di salah satu lokasi wumpus yang mungkin. Ini adalah ditentukan dengan menanyakan di mana $A_{SK}(KB, \neg W_{x,y})$ salah — yaitu, di mana *tidak* diketahui itu ada *tidak* Wumpus a. Fungsi PLAN-SHOT (tidak ditampilkan) menggunakan PLAN-ROUTE untuk merencanakan a urutan tindakan yang akan mengurutkan foto ini. Jika gagal, program mencari kotak jelajahi yang tidak terbukti tidak aman — yaitu, kotak yang $A_{SK}(KB, \neg OK^t_{x,y})$ kembali Salah. Jika tidak ada kotak seperti itu, maka misinya mustahil dan agen mundur ke [1,1] dan memanjat keluar dari gua.

7.7.3 Estimasi kondisi logis

Program agen pada Gambar 7.20 bekerja dengan cukup baik, tetapi ia memiliki satu kelemahan utama: seperti waktu berlalu, biaya komputasi yang terlibat dalam panggilan ke A_{SK} naik dan naik. Ini terjadi terutama karena kesimpulan yang diperlukan harus mundur lebih jauh dan lebih jauh dalam waktu dan melibatkan semakin banyak simbol proposisi. Jelas, ini tidak berkelanjutan — kita tidak bisa memiliki agen yang waktunya memproses setiap persepsi tumbuh sebanding dengan panjang umurnya! Apa yang benar-benar kita butuhkan adalah waktu pembaruan yang *konstan* — yaitu, tidak bergantung pada t. Jawaban yang jelas adalah

CACHING

menyimpan, atau **cache** , hasil inferensi, sehingga proses inferensi pada langkah waktu berikutnya bisa membangun hasil langkah-langkah sebelumnya daripada harus memulai lagi dari awal.

Seperti yang kita lihat di Bagian 4.4, sejarah persepsi masa lalu dan semua konsekuensinya bisa digantikan oleh **negara kepercayaan** —yaitu, beberapa representasi dari himpunan semua arus yang mungkin negara di dunia. [12] Proses memperbarui keadaan kepercayaan saat persepsi baru tiba disebut **estimasi keadaan** . Sedangkan di Bagian 4.4 negara kepercayaan adalah daftar eksplisit negara, di sini kita dapat menggunakan kalimat logis yang melibatkan simbol proposisi yang terkait dengan arus langkah waktu, serta simbol atemporal. Misalnya, kalimat logis

$$\text{WumpusAlive}^1 \wedge \text{L}^1_{2,1} \wedge \text{B}_{2,1} \wedge (\text{P}_{3,1} \vee \text{P}_{2,2}) \tag{7.4}$$

mewakili himpunan semua negara pada waktu 1 di mana wumpus hidup, agennya berada di [2,1], alun-alun itu berangin, dan ada lubang di [3,1] atau [2,2] atau keduanya.

Mempertahankan keadaan keyakinan yang tepat sebagai formula logis ternyata tidak mudah. Jika ada ada n simbol lancar untuk waktu t, maka ada 2^n kemungkinan keadaan — yaitu, penugasan kebenaran nilai untuk simbol-simbol itu. Sekarang, set negara keyakinan adalah set kekuasaan (set semua himpunan bagian) dari mengatur keadaan fisik. Ada 2^n keadaan fisik, maka 2^{2^n} negara keyakinan. Bahkan jika kita menggunakannya penyandian formula logis yang paling ringkas, dengan masing-masing negara kepercayaan terwakili

[12] Kita dapat menganggap persepsi sejarah itu sendiri sebagai representasi dari negara kepercayaan, tetapi yang membuat kesimpulan semakin mahal karena sejarah semakin panjang.

Halaman 37


```
Fungsi HYBRID-WUMPUS-AGENT (persepsi) mengembalikan suatu tindakan
  input : persepsi, daftar, [bau busuk, angin sepoi-sepoi, kilau, benjolan, menjerit]
  gigih : KB, basis pengetahuan, awalnya "fisika wumpus" atemporal
          t, sebuah penghitung, awalnya 0, menunjukkan waktu
          rencana, suatu urutan tindakan, awalnya kosong

  TELL(KB, MAKE-PERCEPT-SENTENCE(persepsi, t))
  TELL the KB kalimat "fisika" temporal untuk waktu t
  safe ← {[x, y]: ASK(KB, OK_{x,y}^t) = true}
  jika ASK(KB, Glitter^t) = true lalu
    plan ← [Grab] + PLAN-ROUTE(saat ini, {[1,1]}, aman) + [Climb]
  jika paket kosong maka
    belum dikunjungi ← {[x, y]: ASK(KB, L_{x,y}^t) = false untuk semua t ≤ t}
    rencana ← PLAN-ROUTE(saat ini, belum dikunjungi ∩ aman, aman)
  jika rencana kosong dan ASK(KB, HaveArrow^t) = true maka
    kemungkinan wumpus ← {[x, y]: ASK(KB, ¬W_{x,y}) = false}
    rencana ← PLAN-SHOT(saat ini, kemungkinan wumpus, aman)
  jika rencana kosong maka // tidak ada pilihan selain mengambil risiko
    tidak tidak aman ← {[x, y]: ASK(KB, ¬OK_{x,y}^t) = salah}
    plan ← PLAN-ROUTE(saat ini, belum dikunjungi ∩ tidak tidak aman, aman)
  jika paket kosong maka
    plan ← PLAN-ROUTE(saat ini, {[1, 1]}, aman) + [Mendaki]
  aksi ← POP(rencana)
  TELL(KB, MAKE-ACTION-SENTENCE(aksi, t))
  t ← t + 1
  kembali tindakan

fungsi PLAN-ROUTE(saat ini, tujuan, diizinkan) mengembalikan urutan tindakan
  input : saat ini, posisi agen saat ini
          tujuan, seperangkat kotak; coba rencanakan rute ke salah satunya
          diizinkan, seperangkat kotak yang dapat membentuk bagian dari rute

  masalah ← ROUTE-PROBLEM(saat ini, sasaran, diizinkan)
  return A*-GRAPH-SEARCH(masalah)
```

Gambar 7.20 Program agen hibrida untuk dunia wumpus. Ini menggunakan pengetahuan proposisional tepi dasar untuk menyimpulkan keadaan dunia, dan kombinasi pencarian pemecahan masalah dan kode khusus domain untuk memutuskan tindakan yang akan diambil.

dengan nomor biner yang unik, kita akan membutuhkan angka dengan $\log_2(2^{2^n}) = 2^n$ bit untuk memberi label kondisi kepercayaan saat ini. Artinya, estimasi keadaan pasti mungkin memerlukan rumus logis yang ukurannya eksponensial dalam jumlah simbol.

Satu skema yang sangat umum dan alami untuk *perkiraan perkiraan* negara adalah untuk mewakili Keyakinan menyatakan sebagai konjungsi literal, yaitu, rumus 1-CNF. Untuk melakukan ini, program agen hanya mencoba untuk membuktikan X^t dan $\neg X^t$ untuk setiap simbol X^t (serta setiap simbol atemporal yang nilai kebenarannya belum diketahui), mengingat status kepercayaan pada t - 1. Konjungsi dari

Halaman 38

Gambar 7.21 Penggambaran status keyakinan 1-CNF (garis tebal) sebagai hanya dapat diwakili, pendekatan konservatif ke keadaan keyakinan (wiggly) yang tepat (wilayah yang diarsir dengan garis putus-putus) garis besar). Setiap dunia yang mungkin ditampilkan sebagai lingkaran; yang teduh konsisten dengan semua persepsi.

literal yang bisa dibuktikan menjadi negara kepercayaan baru, dan negara kepercayaan sebelumnya dibuang.

Penting untuk dipahami bahwa skema ini dapat kehilangan beberapa informasi seiring berjalannya waktu sepanjang. Sebagai contoh, jika kalimat dalam Persamaan (7.4) adalah negara keyakinan yang benar, maka keduanya juga tidak $P_{3,1}$ atau $P_{2,2}$ akan dapat dibuktikan secara individual dan tidak akan muncul dalam keyakinan 1-CNF negara. (Latihan 7.27 mengeksplorasi satu kemungkinan solusi untuk masalah ini.) Di sisi lain, karena setiap literal di negara keyakinan 1-CNF dibuktikan dari negara keyakinan sebelumnya, dan keadaan keyakinan awal adalah pernyataan yang benar, kita tahu bahwa seluruh negara keyakinan 1-CNF harus benar. Dengan demikian, *set negara yang mungkin diwakili oleh negara keyakinan 1-CNF mencakup semua negara yang sebenarnya mungkin diberikan sejarah persept penuh.* Seperti diilustrasikan dalam Gambar 7.21, 1-Keyakinan CNF bertindak sebagai amplop luar sederhana, atau **perkiraan konservatif** , di sekitar keadaan keyakinan yang tepat. Kami melihat ide ini pendekatan konservatif untuk set rumit sebagai tema berulang di banyak bidang AI.

KONSERVATIF PERKIRAAN

7.7.4 Membuat rencana dengan inferensi proposisional

Agen pada Gambar 7.20 menggunakan inferensi logis untuk menentukan kotak mana yang aman, tetapi menggunakan SEBUAH* untuk membuat rencana. Di bagian ini, kami menunjukkan cara membuat rencana dengan inferensi logis. Ide dasarnya sangat sederhana:

1. Bangun kalimat yang mencakup
   (a) Init_0 , kumpulan pernyataan tentang keadaan awal;
   (B) Transisi_1 , ..., Transisi_t , aksioma negara penerus untuk semua tindakan yang mungkin pada setiap waktu hingga t waktu maksimum;
   (c) pernyataan bahwa tujuan tercapai pada saat t: $\text{HaveGold}_t \wedge \text{ClimbedOut}_t$.

Halaman 39

2. Presentasikan seluruh kalimat ke pemecah SAT. Jika pemecah menemukan model yang memuaskan, maka tujuannya dapat dicapai; jika kalimatnya tidak memuaskan, maka masalah perencanaannya adalah mustahil.
3. Dengan asumsi model ditemukan, ekstrak dari model variabel-variabel yang mewakili dan ditugaskan dengan benar. Bersama-sama mereka mewakili rencana untuk mencapai tujuan.

Prosedur perencanaan proposisional, SATP LAN , ditunjukkan pada Gambar 7.22. Ini mengimplementasikan ide dasar yang baru saja diberikan, dengan satu twist. Karena agen tidak tahu berapa langkahnya akan mengambil untuk mencapai tujuan, algoritma mencoba setiap jumlah langkah t yang mungkin, hingga beberapa panjang rencana maksimum yang bisa dipikirkan T_{maks} . Dengan cara ini, dijamin untuk menemukan rencana terpendek jika ada. Karena cara SATP LAN mencari solusi, pendekatan ini tidak bisa digunakan dalam lingkungan yang dapat diamati sebagian; SATP LAN hanya akan mengatur unobservable variabel ke nilai yang dibutuhkan untuk membuat solusi.

fungsi SAT PLAN (init, transisi, tujuan, T_{max}) **mengembalikan** solusi atau kegagalan

```
input : init, transisi, tujuan, merupakan deskripsi masalah
        T maks , batas atas untuk panjang rencana

untuk t = 0 hingga T max do
  cnf ← T RANSLATE -T O -SAT (init, transisi, goal, t)
  model ← SAT-S OLVER (cnf)
  jika model tidak nol maka
    return E XTRACT -S OLUTION (model)
kegagalan pengembalian
```

Gambar 7.22 Algoritma SATP LAN . Masalah perencanaan diterjemahkan ke dalam CNF kalimat di mana tujuan tersebut dinyatakan berlaku pada langkah waktu t tetap dan aksioma dimasukkan untuk setiap kali naik ke t. Jika algoritma satisfiability menemukan model, maka sebuah rencana diekstraksi dengan melihat simbol-simbol proposisi yang merujuk pada tindakan dan ditugaskan benar dalam model. Jika tidak ada model, maka proses diulangi dengan tujuan dipindahkan satu langkah kemudian.

Langkah kunci dalam menggunakan SATP LAN adalah pembangunan basis pengetahuan. Itu mungkin tampaknya, pada inspeksi biasa, bahwa aksioma dunia wumpus dalam Bagian 7.7.1 cukup untuk langkah-langkah 1 (a) dan 1 (b) di atas. Namun, ada perbedaan yang signifikan antara persyaratan untuk entailment (seperti yang diuji oleh A SK) dan yang untuk kepuasan. Pertimbangkan, misalnya, agen lokasi, awalnya [1,1], dan anggap tujuan ambisius agen adalah berada di [2,1] pada waktu 1. Basis pengetahuan awal berisi $L^0_{1,1}$ dan tujuannya adalah $L^1_{2,1}$. Dengan menggunakan SK , kita dapat membuktikan $L^1_{2,1}$ jika *Forward* 0 dinyatakan, dan, meyakinkan, kami tidak dapat membuktikan $L^1_{2,1}$ jika, katakanlah, *Tembak* 0 dinyatakan sebagai gantinya. Sekarang, SATP LAN akan menemukan rencana [*Maju* 0]; sejauh ini bagus. Sayangnya, SATP LAN juga menemukan rencana [*Tembak* 0]. Bagaimana ini bisa terjadi? Untuk mengetahuinya, kami memeriksa model yang SATP LAN konstruksi: itu termasuk tugas $L^0_{2,1}$, yaitu agen bisa berada di [2,1] pada waktu 1 dengan berada di sana pada waktu 0 dan menembak. Orang mungkin bertanya, “Bukankah kita bilang agennya masuk? [1,1] pada waktu 0? ”Ya, kami melakukannya, tetapi kami tidak memberi tahu agen itu bahwa itu tidak dapat berada di dua tempat sekaligus! Untuk entailment, $L^0_{2,1}$ tidak diketahui dan oleh karena itu, tidak dapat digunakan dalam bukti; untuk kepuasan,

Halaman 40

di sisi lain, $L^0_{2,1}$ tidak diketahui dan oleh karena itu, dapat diatur ke nilai apa pun yang membantu mewujudkan tujuan. Untuk alasan ini, SATP LAN adalah alat debugging yang baik untuk basis pengetahuan karena itu mengungkapkan tempat-tempat di mana pengetahuan tidak ada. Dalam kasus khusus ini, kita dapat memperbaiki basis pengetahuan dengan menyatakan bahwa, pada setiap langkah waktu, agen tepat berada di satu lokasi, menggunakan kumpulan kalimat yang mirip dengan yang digunakan untuk menegaskan keberadaan persis satu wumpus. Atau, kita dapat menyatakan $\neg L^0_{x,y}$ untuk semua lokasi selain [1,1]; aksioma negara penerus untuk lokasi mengurus langkah waktu berikutnya. Perbaikan yang sama juga berfungsi untuk memastikan agen hanya memiliki satu orientasi.

Namun, SATP LAN memiliki lebih banyak kejutan. Yang pertama adalah menemukan model tindakan mustahil, seperti menembak tanpa panah. Untuk memahami alasannya, kita perlu melihat lebih banyak hati-hati pada apa aksioma negara penerus (seperti Persamaan (7.3)) katakan tentang tindakan yang prasyarat tidak terpenuhi. Aksioma *memang* memprediksi dengan benar bahwa tidak ada yang akan terjadi kapan tindakan semacam itu dilakukan (lihat Latihan 10.14), tetapi mereka *tidak* mengatakan bahwa tindakan itu tidak mungkin dieksekusi! Untuk menghindari membuat rencana dengan tindakan ilegal, kita harus menambahkan **aksioma prakondisi** menyatakan bahwa suatu kejadian tindakan membutuhkan prasyarat untuk dipenuhi. [13] Misalnya, kita perlu mengatakan, untuk setiap kali t, itu

PRASYARAT AXIOMS

$$\text{Menembak}^t \Rightarrow \text{HaveArrow}^t .$$

Ini memastikan bahwa jika suatu rencana memilih aksi *Tembak* kapan saja, itu harus menjadi kasus bahwa agen memiliki panah pada saat itu.

Kejutan kedua SATP LAN adalah penciptaan rencana dengan beberapa tindakan simultan. Misalnya, mungkin muncul dengan model di mana Forward 0 dan Shoot 0 benar, yang tidak diizinkan. Untuk menghilangkan masalah ini, kami memperkenalkan **aksioma pengecualian tindakan** : untuk setiap pasangan tindakan A^t_i dan A^t_j kita tambahkan aksioma

PENGECUALIAN TINDAKAN AKSIOMA

$$\neg A^t_i \vee \neg A^t_j .$$

Mungkin ditunjukkan bahwa berjalan maju dan menembak pada saat yang sama tidak terlalu sulit untuk dilakukan

lakukan, sedangkan, katakanlah, menembak dan meraih pada saat yang sama agak tidak praktis. Dengan memaksakan aksioma pengecualian tindakan hanya pada pasangan tindakan yang benar-benar saling mengganggu, kami dapat memungkinkan untuk rencana yang menyertakan beberapa tindakan simultan — dan karena SATP LAN ditemukan rencana hukum terpendek, kita dapat yakin bahwa itu akan mengambil keuntungan dari kemampuan ini.

Untuk meringkas, SATP LAN menemukan model untuk kalimat yang berisi keadaan awal, the tujuan, aksioma negara penerus, aksioma prakondisi, dan aksioma tindakan pengucilan. Dapat ditunjukkan bahwa kumpulan aksioma ini cukup, dalam arti tidak ada lagi "solusi" palsu. Model apa pun yang memenuhi hukuman proposisional adalah a rencana yang valid untuk masalah asli. Teknologi pemecahan SAT modern membuat pendekatan cukup praktis. Misalnya, pemecah gaya DPLL tidak mengalami kesulitan dalam menghasilkan 11 langkah solusi untuk instance dunia wumpus yang ditunjukkan pada Gambar 7.2.

Bagian ini menjelaskan pendekatan deklaratif untuk konstruksi agen: agen bekerja dengan kombinasi kalimat tegas dalam basis pengetahuan dan melakukan kesimpulan logis ence. Pendekatan ini memiliki beberapa kelemahan yang tersembunyi dalam frasa seperti "untuk setiap kali t" dan

[13] Perhatikan bahwa penambahan aksioma prakondisi berarti bahwa kita tidak perlu memasukkan prasyarat untuk tindakan dalam aksioma negara penerus.

Halaman 41

"Untuk setiap kuadrat [x, y]." Untuk setiap agen praktis, frasa ini harus diimplementasikan oleh kode yang menghasilkan contoh skema kalimat umum secara otomatis untuk dimasukkan ke dalam basis pengetahuan. Untuk dunia wumpus dengan ukuran yang masuk akal — yang sebanding dengan yang kecil permainan komputer — kita mungkin membutuhkan papan 100 × 100 dan 1000 langkah waktu, yang mengarah ke pengetahuan basis tepi dengan puluhan atau ratusan juta kalimat. Ini tidak hanya menjadi lebih tidak praktis, tetapi juga menggambarkan masalah yang lebih dalam: kita tahu sesuatu tentang dunia pus — yaitu, bahwa "fisika" bekerja dengan cara yang sama di semua kotak dan sepanjang waktu langkah-langkah — yang tidak dapat kita ungkapkan secara langsung dalam bahasa logika proposisional. Untuk mengatasi ini masalah, kita membutuhkan bahasa yang lebih ekspresif, di mana frasa seperti "untuk setiap kali t" dan "untuk setiap kuadrat [x, y]" dapat ditulis dengan cara alami. Logika orde pertama, dijelaskan dalam Bab 8, adalah bahasa seperti itu; dalam logika orde pertama dunia wumpus dengan berbagai ukuran dan durasi dapat dijelaskan dalam sekitar sepuluh kalimat daripada sepuluh juta atau sepuluh triliun.

7.8 S RINGKASAN

Kami telah memperkenalkan agen berbasis pengetahuan dan telah menunjukkan cara mendefinisikan logika dengan yang agen-agen semacam itu dapat alasan tentang dunia. Poin utamanya adalah sebagai berikut:

- Agen cerdas membutuhkan pengetahuan tentang dunia untuk mencapai keputusan yang baik.
- Pengetahuan terkandung dalam agen dalam bentuk **kalimat** dalam **representasi pengetahuan bahasa** yang disimpan dalam **basis pengetahuan** .
- Agen berbasis pengetahuan terdiri dari basis pengetahuan dan mekanisme inferensi nism. Ini beroperasi dengan menyimpan kalimat tentang dunia dalam basis pengetahuannya, menggunakan mekanisme inferensi untuk menyimpulkan kalimat baru, dan menggunakan kalimat ini untuk memutuskan apa tindakan yang harus diambil.
- Bahasa representasi didefinisikan oleh **sintaksnya** , yang menentukan struktur kalimat, dan **semantiknya** , yang mendefinisikan **kebenaran** setiap kalimat dalam setiap **kemungkinan dunia** atau **model** .
- Hubungan **entailment** antara kalimat sangat penting untuk pemahaman kita tentang pemikiran. Sebuah kalimat α mencakup kalimat lain β jika β benar di semua dunia di mana α benar. Definisi Setara mencakup **validitas** kalimat $\alpha \Rightarrow \beta$ dan **ketidakpuasan** kalimat $\alpha \wedge \neg\beta$.
- Inferensi adalah proses memperoleh kalimat baru dari yang lama. Kesimpulan inferensi **suara** rithms *hanya* menghasilkan kalimat yang disyaratkan; algoritme **lengkap** mendapatkan *semua* kalimat yang disyaratkan.
- **Logika proposisional** adalah bahasa sederhana yang terdiri dari **simbol** - **simbol proposisi** dan **logika penghubung** . Itu dapat menangani proposisi yang dikenal benar, dikenal salah, atau sepenuhnya

Tidak dikenal

• Seperangkat model yang mungkin, diberi kosa kata proposisional tetap, terbatas, jadi tailment dapat diperiksa dengan menghitung model. Inferensi **pengecekan model yang** efisien algoritma untuk logika proposisional termasuk metode penelusuran mundur dan pencarian lokal dan seringkali dapat memecahkan masalah besar dengan cepat.

Halaman 42

• **Aturan inferensi** adalah pola inferensi suara yang dapat digunakan untuk menemukan bukti. Itu aturan **resolusi** menghasilkan algoritma inferensi lengkap untuk basis pengetahuan yang dinyatakan dalam **bentuk normal konjungtif** . **Penerusan ke depan** dan **ke belakang** adalah algoritma penalaran yang sangat alami untuk basis pengetahuan dalam **bentuk Horn** .
• Metode **pencarian lokal** seperti W ALK SAT dapat digunakan untuk menemukan solusi. Seperti Ritme baik tetapi tidak lengkap.
• **Estimasi kondisi** logis melibatkan pemeliharaan kalimat logis yang menggambarkan himpunan dari kemungkinan keadaan yang konsisten dengan sejarah pengamatan. Setiap langkah pembaruan membutuhkan inferensi menggunakan model transisi lingkungan, yang dibangun dari **penerus nyatakan aksioma** yang menentukan bagaimana masing-masing **fasih** berubah.
• Keputusan dalam agen logis dapat dibuat dengan pemecahan SAT: menemukan model yang mungkin menentukan urutan tindakan masa depan yang mencapai tujuan. Pendekatan ini hanya berfungsi untuk sepenuhnya dapat diamati atau lingkungan tanpa sensor.
• Logika proposisional tidak menskala ke lingkungan dengan ukuran tidak terbatas karena kurang kekuatan ekspresif untuk berurusan secara singkat dengan waktu, ruang, dan pola universal dari hubungan ikatan antar objek.

B IBLIOGRAPHICAL DAN H ISTORICAL N OTES

Makalah John McCarthy "Program dengan Common Sense" (McCarthy, 1958, 1968) mengumumkan gated gagasan agen yang menggunakan penalaran logis untuk memediasi antara persepsi dan tindakan. Itu juga mengangkat bendera deklarativisme, menunjukkan bahwa memberi tahu agen apa yang perlu diketahui adalah cara yang elegan untuk membangun perangkat lunak. Artikel Allen Newell (1982) "Tingkat Pengetahuan" membuat kasus bahwa agen rasional dapat dijelaskan dan dianalisis pada tingkat abstrak yang ditentukan oleh pengetahuan yang mereka miliki daripada program yang mereka jalankan. Deklaratif dan prosedur pendekatan dural terhadap AI dianalisis secara mendalam oleh Boden (1977). Debat dihidupkan kembali oleh, antara lain, Brooks (1991) dan Nilsson (1991), dan berlanjut hingga hari ini (Shaparau *et al.* , 2008). Sementara itu, pendekatan deklaratif telah menyebar ke bidang ilmu komputer lainnya seperti jaringan (Loo *et al.* , 2006).

Logika sendiri berawal dari filsafat dan matematika Yunani kuno. Berbagai log- prinsip ical — prinsip yang menghubungkan struktur sintaksis kalimat dengan kebenarannya dan kepalsuan, dengan maknanya, atau dengan validitas argumen yang mereka pikirkan — adalah tersebar di karya-karya Plato. Studi sistematis logis pertama yang diketahui dilakukan oleh Aristoteles, yang karyanya dirakit oleh murid-muridnya setelah kematiannya pada 322 B . C . sebagai risalah yang disebut *Organon* . **Silogisme** Aristoteles adalah apa yang sekarang kita sebut inferensi aturan Meskipun silogisme termasuk unsur-unsur baik proposisional dan logika tingkat pertama, sistem secara keseluruhan tidak memiliki sifat komposisi yang diperlukan untuk menangani kalimat kompleksitas yang berubah-ubah.

SILOGISME

Sekolah-sekolah Megarian dan Stoic terkait erat (berasal pada abad kelima B . C . dan berlanjut selama beberapa abad sesudahnya) memulai studi sistematis dari logika dasar penghubung. Penggunaan tabel kebenaran untuk mendefinisikan penghubung adalah karena Philo dari Megara. Itu

Halaman 43

Stoics mengambil lima aturan inferensi dasar sebagai valid tanpa bukti, termasuk aturan yang sekarang kita panggil Modus Ponens. Mereka menurunkan sejumlah aturan lain dari kelima aturan ini, menggunakan, antara lain prinsip, teorema pengurang (halaman 249) dan jauh lebih jelas tentang gagasan bukti daripada Aristoteles. Akun yang baik tentang sejarah logika Megarian dan Stoic diberikan oleh Benson Mates (1953).

Gagasan mengurangi inferensi logis pada proses mekanis murni yang diterapkan pada Bahasa Mal adalah karena Wilhelm Leibniz (1646-1716), meskipun ia memiliki keberhasilan yang terbatas dalam meningkatkan menerapkan ide-ide. George Boole (1847) memperkenalkan yang komprehensif pertama dan bisa diterapkan sistem logika formal dalam bukunya *The Mathematical Analysis of Logic* . Logika Boole adalah dimodelkan dengan ketat pada aljabar biasa bilangan real dan menggunakan substitusi secara logis ekspresi setara sebagai metode inferensi utamanya. Meskipun sistem Boole masih jatuh singkat dari logika proposisional penuh, itu cukup dekat sehingga matematikawan lain dapat dengan cepat isi celahnya. Schröder (1877) menggambarkan bentuk normal konjungtif, sedangkan bentuk Horn adalah diperkenalkan kemudian oleh Alfred Horn (1951). Eksposisi komprehensif pertama modern logika proposisional (dan logika tingkat pertama) ditemukan dalam *Begriffschrift* Gottlob Frege (1879) ("Menulis Konsep" atau "Notasi Konseptual").

Perangkat mekanis pertama yang melakukan inferensi logis dibangun oleh yang ketiga Earl of Stanhope (1753–1816). Demonstrator Stanhope dapat menangani silogisme dan kesimpulan tertentu dalam teori probabilitas. William Stanley Jevons, salah satunya ditingkatkan dan diperpanjang karya Boole, membangun "piano logis" pada tahun 1869 untuk melakukan membentuk inferensi dalam logika Boolean. Sejarah yang menghibur dan instruktif dari ini dan lainnya perangkat mekanik awal untuk alasan diberikan oleh Martin Gardner (1968). Pub- pertama Program komputer yang menarik untuk inferensi logis adalah Logic Theorist of Newell, Shaw, dan Simon (1957). Program ini dimaksudkan untuk memodelkan proses pemikiran manusia. Merusak- tin Davis (1957) sebenarnya merancang sebuah program yang muncul dengan bukti pada tahun 1954, tetapi Hasil Teori Logika diterbitkan sedikit lebih awal.

Tabel kebenaran sebagai metode pengujian validitas atau ketidakpuasan dalam logika proposisional adalah diperkenalkan secara independen oleh Emil Post (1921) dan Ludwig Wittgenstein (1922). Pada 1930-an, banyak kemajuan dibuat pada metode inferensi untuk logika tingkat pertama. Khususnya, Gödel (1930) menunjukkan bahwa prosedur lengkap untuk inferensi dalam logika orde pertama bisa dilakukan diperoleh melalui pengurangan logika proposisional, menggunakan teorema Herbrand (Herbrand, 1930). Kita mengambil sejarah ini lagi di Bab 9; Poin penting di sini adalah pengembangan algoritma proposisi efisien di tahun 1960 - an sebagian besar dimotivasi oleh kepentingan matematikawan dalam teorema teorema yang efektif untuk logika tingkat pertama. The Davis – Putnam juga rithm (Davis dan Putnam, 1960) adalah algoritma efektif pertama untuk resolusi proposisional tetapi dalam banyak kasus jauh lebih efisien daripada algoritma backtracking DPLL yang diperkenalkan dua tahun kemudian (1962). Aturan resolusi penuh dan bukti kelengkapannya muncul di a makalah seminal oleh JA Robinson (1965), yang juga menunjukkan bagaimana melakukan penalaran tingkat pertama tanpa menggunakan teknik proposisional.

Stephen Cook (1971) menunjukkan bahwa menentukan kepuasan suatu kalimat secara proposisional logika (masalah SAT) adalah NP-complete. Karena memutuskan entailment sama dengan decid- ing ketidakpuasan, itu co-NP-lengkap. Banyak himpunan bagian dari logika proposisional dikenal yang masalah kepuasannya dapat dipecahkan secara polinomi; Klausa klakson adalah salah satu bagiannya.

Halaman 44

Algoritma forward-chaining linear-time untuk klausa Horn disebabkan oleh Dowling dan Gallier (1984), yang menggambarkan algoritma mereka sebagai proses aliran data yang mirip dengan propagasi sinyal nals dalam suatu rangkaian.

Investigasi teoritis awal menunjukkan bahwa DPLL memiliki komposisi kasus rata-rata polinomial

kerumitan untuk distribusi alami masalah tertentu. Fakta yang berpotensi menarik ini menjadi kurang menggairahkan ketika Franco dan Paull (1983) menunjukkan bahwa masalah yang sama dapat diselesaikan dalam waktu yang konstan hanya dengan menebak tugas acak. Metode generasi acak dijelaskan dalam bab ini menghasilkan masalah yang jauh lebih sulit. Termotivasi oleh kesuksesan empiris pencarian lokal pada masalah ini, Koutsoupias dan Papadimitriou (1992) menunjukkan bahwa Algoritme pendakian bukit dapat menyelesaikan *hampir semua* contoh masalah kepuasan dengan sangat cepat, menunjukkan bahwa masalah sulit jarang terjadi. Selain itu, Schöning (1999) menunjukkan secara acak algoritma hill-climbing yang *kasus terburuknya* diperkirakan run time pada masalah 3-SAT (yaitu, satisfiability dari kalimat 3-CNF) adalah O (1,333 ^{n}) —sangat eksponensial, tetapi jauh lebih cepat daripada batas kasus terburuk sebelumnya. Rekor saat ini adalah O (1,324 ^{n}) (Iwama dan Tamaki, 2004). Achlioptas *et al.* (2004) dan Alekhnovich *et al.* (2005) menunjukkan keluarga contoh 3-SAT dimana semua algoritma yang dikenal seperti DPLL membutuhkan waktu berjalan eksponensial.

Di sisi praktis, keuntungan efisiensi dalam pemecah proposisional telah ditandai. Diberikan sepuluh menit waktu komputasi, algoritma DPLL asli pada tahun 1962 hanya bisa menyelesaikan masalah kelihatannya tidak lebih dari 10 atau 15 variabel. Pada 1995 pemecah S ATZ (Li dan Anbulagan, 1997) dapat menangani 1.000 variabel, berkat struktur data yang dioptimalkan untuk pengindeksan variabel ables Dua kontribusi penting adalah teknik pengindeksan **harfiah yang diamati** dari Zhang dan Stickel (1996), yang membuat perambatan unit sangat efisien, dan pengenalan klausa (yaitu, kendala) teknik pembelajaran dari komunitas CSP oleh Bayardo dan Schrag (1997). Menggunakan ide-ide ini, dan didorong oleh prospek penyelesaian verifikasi sirkuit skala industri masalah, Moskewicz *et al.* (2001) mengembangkan pemecah C HAFF , yang dapat menangani masalah penuh dengan jutaan variabel. Mulai tahun 2002, kompetisi SAT telah diadakan berliku-liku; sebagian besar entri yang menang adalah keturunan C HAFF atau telah menggunakan pendekatan umum yang sama. RS AT (Pipatsrisawat dan Darwiche, 2007), pemenang tahun 2007, jatuh pada kategori terakhir. Yang juga patut diperhatikan adalah M INI SAT (Een dan Sörensson, 2003), sebuah open-source implementasi tersedia di http://minisat.se yang dirancang agar mudah dimodifikasi dan ditingkatkan. Lansekap pemecah saat ini disurvei oleh Gomes *et al.* (2008).

Algoritma pencarian lokal untuk kepuasan telah dicoba oleh berbagai penulis di seluruh 1980-an; semua algoritma didasarkan pada ide meminimalkan jumlah yang tidak puas klausa (Hansen dan Jaumard, 1990). Algoritma yang sangat efektif dikembangkan oleh Gu (1989) dan secara mandiri oleh Selman *et al.* (1992), yang menyebutnya GSAT dan menunjukkan hal itu itu mampu memecahkan berbagai masalah yang sangat sulit dengan sangat cepat. W ALK SAT algoritma yang dijelaskan dalam bab ini disebabkan oleh Selman *et al.* (1996).

"Transisi fase" dalam pemenuhan masalah k-SAT acak pertama kali diamati oleh Simon dan Dubois (1989) dan telah memunculkan banyak teori dan empiris penelitian — sebagian karena hubungan yang jelas dengan fenomena transisi fase dalam statistik fisika. Cheeseman *et al.* (1991) mengamati transisi fase dalam beberapa CSP dan dugaan bahwa semua masalah NP-hard memiliki transisi fase. Crawford dan Auton (1993) menemukan Transisi 3-SAT pada rasio klausa / variabel sekitar 4,26, mencatat bahwa ini bertepatan dengan a

Halaman 45

puncak yang tajam dalam waktu menjalankan pemecah SAT mereka. Cook dan Mitchell (1997) memberikan yang sangat baik ringkasan literatur awal tentang masalah tersebut.

Keadaan pemahaman teoritis saat ini diringkas oleh Achlioptas (2009).

KEPUASAN AMBANG DUGAAN

The **ambang satisfiability dugaan** menyatakan bahwa, untuk setiap k, ada satisfiability tajam ambang batas r_k , sedemikian sehingga jumlah variabel $n \to \infty$, contoh di bawah ambang batas adalah *memuaskan* dengan probabilitas 1, sedangkan yang di atas ambang tidak *memuaskan* dengan probabilitas bility 1. Dugaan itu tidak cukup dibuktikan oleh Friedgut (1999): ada ambang yang tajam tetapi lokasinya mungkin bergantung pada n bahkan $n \to \infty$. Meskipun ada kemajuan yang signifikan dalam asimptotik analisis lokasi ambang batas untuk k besar (Achlioptas dan Peres, 2004; Achlioptas *et al.* , 2007), semua yang dapat dibuktikan untuk k = 3 adalah bahwa ia terletak pada kisaran [3,52,4,51]. Teori saat ini menunjukkan bahwa puncak dalam run time dari pemecah SAT tidak selalu terkait dengan kepuasan ambang batas bility, tetapi alih-alih ke fase transisi dalam distribusi solusi dan struktur Instance SAT. Hasil empiris karena Coarfa *et al.* (2003) mendukung pandangan ini. Bahkan,

SURVEI PERAMBATAN

goritme seperti **propagasi survei** (Parisi dan Zecchina, 2002; Maneva *et al.* , 2007) ambil keuntungan dari properti khusus instance SAT acak di dekat ambang kepuasan dan sangat mengungguli pemecah SAT umum pada contoh seperti itu.

Sumber terbaik untuk informasi tentang kepuasan, baik teoretis maupun praktis, adalah *Buku Pegangan Kepuasan* (Biere *et al.* , 2009) dan *Konferensi Internasional* reguler *di Teori dan Aplikasi Pengujian Kepuasan* , dikenal sebagai SAT.

Gagasan agen bangunan dengan logika proposisional dapat ditelusuri kembali ke mani makalah McCulloch dan Pitts (1943), yang memprakarsai bidang jaringan saraf. Menipu- bertentangan dengan anggapan populer, makalah ini berkaitan dengan implementasi Boolean desain agen berbasis sirkuit di otak. Agen berbasis sirkuit, yang melakukan perhitungan oleh menyebarkan sinyal di sirkuit perangkat keras daripada menjalankan algoritma untuk keperluan umum komputer, telah menerima sedikit perhatian dalam AI, namun. Pengecualian yang paling menonjol adalah karya Stan Rosenschein (Rosenschein, 1985; Kaelbling dan Rosenschein, 1990), yang mendesain mengembangkan cara untuk mengkompilasi agen berbasis sirkuit dari deskripsi deklaratif dari tugas lingkungan ronment. (Pendekatan Rosenschein dijelaskan secara panjang lebar dalam edisi kedua ini buku.) Karya Rod Brooks (1986, 1989) menunjukkan efektivitas berbasis sirkuit desain untuk mengendalikan robot — topik yang kita bahas dalam Bab 25. Brooks (1991) berpendapat bahwa desain berbasis sirkuit adalah *semua* yang diperlukan untuk AI — representasi dan alasan itu rumit, mahal, dan tidak perlu. Dalam pandangan kami, tidak ada pendekatan yang cukup oleh diri. Williams *et al.* (2003) menunjukkan bagaimana desain agen hibrida tidak terlalu berbeda dari kami agen wumpus telah digunakan untuk mengendalikan pesawat ruang angkasa NASA, merencanakan urutan tindakan dan mendiagnosis dan pulih dari kesalahan.

Masalah umum dalam melacak lingkungan yang dapat diamati sebagian adalah direduksi untuk representasi berbasis negara dalam Bab 4. Instansiasinya untuk perwakilan perwakilan Sentimen dipelajari oleh Amir dan Russell (2003), yang mengidentifikasi beberapa kelas lingkungan ronments yang mengakui algoritma estimasi negara yang efisien dan menunjukkan bahwa untuk beberapa lainnya kelas masalahnya tidak bisa dipecahkan. Masalah **proyeksi-temporal** , yang melibatkan penentuan menambang proposisi apa yang benar setelah urutan tindakan dijalankan, dapat dilihat sebagai a kasus khusus estimasi negara dengan persepsi kosong. Banyak penulis telah mempelajari masalah ini karena pentingnya dalam perencanaan; beberapa hasil kekerasan penting ditetapkan oleh

SEMENTARA-PROYEKSI

Halaman 46

Liberatore (1997). Gagasan untuk mewakili negara keyakinan dengan proposisi dapat dilacak Wittgenstein (1922).

Estimasi keadaan logis, tentu saja, membutuhkan representasi logis dari efek aksi — masalah utama dalam AI sejak akhir 1950-an. Usulan yang dominan telah **sit** formalisme **kalkulus** (McCarthy, 1963), yang ditulis dalam logika tingkat pertama. Kita mendiskusikan kalkulus situasi, dan berbagai ekstensi dan alternatif, dalam Bab 10 dan 12. The pendekatan yang diambil dalam bab ini — menggunakan indeks temporal pada variabel proposisional — lebih dari itu membatasi tetapi memiliki manfaat kesederhanaan. Pendekatan umum yang terkandung dalam LAN SATP Algoritma ini diusulkan oleh Kautz dan Selman (1992). Generasi selanjutnya dari SATP LAN adalah mampu mengambil keuntungan dari kemajuan dalam pemecah SAT, dijelaskan sebelumnya, dan tetap di antara cara paling efektif untuk menyelesaikan masalah yang sulit (Kautz, 2006).

Masalah **bingkai** pertama kali diakui oleh McCarthy dan Hayes (1969). Banyak pencari menganggap masalah tidak terpecahkan dalam logika tingkat pertama, dan itu memacu hebat menangani penelitian logika nonmonotonik. Filsuf dari Dreyfus (1972) ke Crockett (1994) telah mengutip masalah kerangka sebagai salah satu gejala kegagalan yang tak terelakkan dari keseluruhan Perusahaan AI. Solusi dari masalah bingkai dengan aksioma state-successor adalah karena Ray Reiter (1991). Thielscher (1999) mengidentifikasi masalah kerangka inferensial sebagai ide terpisah dan memberikan solusi. Dalam retrospeksi, orang dapat melihat bahwa agen Rosenschein (1985) adalah menggunakan sirkuit yang menerapkan aksioma negara penerus, tetapi Rosenschein tidak menyadarinya masalah bingkai dengan demikian sebagian besar diselesaikan. Foo (2001) menjelaskan mengapa peristiwa diskrit model teori kontrol yang biasanya digunakan oleh para insinyur tidak harus secara eksplisit berurusan dengan bingkai masalah: karena mereka berurusan dengan prediksi dan kontrol, bukan dengan penjelasan dan alasan tentang situasi kontrafaktual.

Pemecah proposisional modern memiliki penerapan yang luas dalam aplikasi industri. Ap- Penerapan inferensi proposisional dalam sintesis perangkat keras komputer sekarang menjadi standar teknik memiliki banyak penyebaran skala besar (Nowick *et al.* , 1993). SATMC memuaskan pemeriksa kemampuan digunakan untuk mendeteksi kerentanan yang sebelumnya tidak diketahui pada pengguna browser Web protokol masuk (Armando *et al.* , 2008).

Dunia wumpus diciptakan oleh Gregory Yob (1975). Ironisnya, Yob mengembangkannya karena dia bosan dengan permainan yang dimainkan di kotak persegi panjang: topologi aslinya dunia wumpus adalah sebuah dodecahedron, dan kami memasukkannya kembali ke dalam kisi lama yang membosankan. Michael Genesereth adalah orang pertama yang menyarankan bahwa dunia wumpus digunakan sebagai agen yang diuji.

E XERCISES

7.1 Misalkan agen telah berkembang ke titik yang ditunjukkan pada Gambar 7.4 (a), halaman 239, memiliki tidak merasakan apa pun di [1,1], angin sepoi-sepoi di [2,1], dan bau busuk di [1,2], dan sekarang memusatkan perhatian pada isi [1,3], [2,2], dan [3,1]. Masing-masing dapat berisi lubang, dan paling banyak satu bisa mengandung wumpus. Mengikuti contoh Gambar 7.5, buat himpunan dunia yang mungkin. (Anda harus menemukan 32 di antaranya.) Tandai dunia di mana KB benar dan dunia di mana

Halaman 47

masing-masing kalimat berikut ini benar:

α_2 = "Tidak ada lubang di [2,2]."
α_3 = "Ada wumpus di [1,3]."

Oleh karena itu tunjukkan bahwa KB $\models \alpha_2$ dan KB $\models \alpha_3$.

7.2 (Diadaptasi dari Barwise dan Etchemendy (1993).) Diberikan hal berikut, dapatkah Anda membuktikan bahwa unicorn itu mitos? Bagaimana dengan sihir? Bertanduk?

Jika unicorn adalah mitos, maka itu abadi, tetapi jika itu tidak mitos, maka itu adalah a mamalia fana. Jika unicorn itu abadi atau mamalia, maka itu bertanduk.
Unicorn itu ajaib jika bertanduk.

7.3 Pertimbangkan masalah memutuskan apakah kalimat logika proposisional benar dalam a model yang diberikan.

a . Tulis algoritma rekursif PL-T RUE ? (S, m) yang mengembalikan true jika dan hanya jika sen- t s s benar dalam model m (di mana m memberikan nilai kebenaran untuk setiap simbol dalam s). Algoritme harus berjalan dalam waktu linier dalam ukuran kalimat. (Atau, gunakan a versi fungsi ini dari repositori kode online.)

b . Berikan tiga contoh kalimat yang bisa ditentukan benar atau salah secara *parsial* model yang tidak menentukan nilai kebenaran untuk beberapa simbol.

c . Tunjukkan bahwa nilai kebenaran (jika ada) dari suatu kalimat dalam model parsial tidak dapat ditentukan efisien secara umum.

d . Ubah PL-T RUE Anda ? algoritma sehingga kadang-kadang bisa menilai kebenaran dari parsial model, sambil mempertahankan struktur rekursif dan jangka waktu liniernya. Berikan tiga contoh kalimat yang kebenarannya dalam model parsial *tidak* terdeteksi oleh algoritma Anda.

e . Selidiki apakah algoritma yang dimodifikasi membuat TT-E NTAILS ? lebih efisien.

7.4 Manakah dari berikut ini yang benar?

a . Salah $\models$ Benar.

b . Benar $\models$ Salah.

c . $(A \wedge B) \models (A \Leftrightarrow B)$.

d . $A \Leftrightarrow B \models A \vee B$.

e . $A \Leftrightarrow B \models \neg A \vee B$.

f . $(A \wedge B) \Rightarrow C \models (A \Rightarrow C) \vee (B \Rightarrow C)$.

g . $(C \vee (\neg A \wedge \neg B)) \equiv ((A \Rightarrow C) \wedge (B \Rightarrow C))$.

h . $(A \vee B) \wedge (\neg C \vee \neg D \vee E) \models (A \vee B)$.

i . $(A \vee B) \wedge (\neg C \vee \neg D \vee E) \models (A \vee B) \wedge (\neg D \vee E)$.

j . $(A \vee B) \wedge \neg (A \Rightarrow B)$ memuaskan.

k . (A ⇔ B) ∧ (¬A ∨ B) memuaskan.

l . (A ⇔ B) ⇔ C memiliki jumlah model yang sama dengan (A ⇔ B) untuk setiap set tetap simbol proposisi yang mencakup A, B, C.

Halaman 48

7.5 Buktikan masing-masing pernyataan berikut:

a . α valid jika dan hanya jika True | = α.

b . Untuk α, False | = α.

c . α | = β jika dan hanya jika kalimat (α ⇒ β) valid.

d . α ≡ β jika dan hanya jika kalimat (α ⇔ β) valid.

e . α | = β jika dan hanya jika kalimat (α ∧ ¬β) tidak memuaskan.

7.6 Buktikan, atau temukan contoh tandingan untuk, masing-masing pernyataan berikut:

a . Jika α | = γ atau β | = γ (atau keduanya) maka (α ∧ β) | = γ

b . Jika α | = (β ∧ γ) maka α | = β dan α | = γ.

c . Jika α | = (β ∨ γ) maka α | = β atau α | = γ (atau keduanya).

7.7 Pertimbangkan kosa kata dengan hanya empat proposisi, A, B, C, dan D. Berapa banyak model apakah ada kalimat berikut?

a . B ∨ C.

b . ¬A ∨ ¬B C ¬C ∨ ¬D.

c . (A ⇒ B) ∧ A ∧ ¬B ∧ C ∧ D.

7.8 Kami telah mendefinisikan empat penghubung logis biner.

a . Apakah ada orang lain yang mungkin berguna?

b . Ada berapa banyak koneksi biner?

c . Mengapa beberapa di antaranya tidak terlalu berguna?

7.9 Menggunakan metode pilihan Anda, verifikasi masing-masing persamaan pada Gambar 7.11 (halaman 249).

7.10 Putuskan apakah masing-masing kalimat berikut ini valid, tidak memuaskan, atau tidak. Verjika keputusan Anda menggunakan tabel kebenaran atau aturan ekivalensi pada Gambar 7.11 (halaman 249).

a . Asap ⇒ Asap

b . Asap ⇒ Api

c . (Asap ⇒ Api) ⇒ (¬Merokok ⇒ Api)

d . Asap ∨ Api ∨ ¬ Api

e . ((Asap ∧ Panas) ⇒ Api) ⇔ ((Asap ⇒ Api) ∨ (Panas ⇒ Api))

f . (Asap ⇒ Api) ⇒ ((Asap ∧ Panas) ⇒ Api)

g . Big ∨ Dumb ∨ (Big ⇒ Dumb)

7.11 Setiap kalimat logika proposisional secara logis setara dengan pernyataan bahwa setiap posisi dunia yang adil di mana itu akan salah tidak demikian. Dari pengamatan ini, buktikan bahwa ada Kalimat dapat ditulis dalam CNF.

7.12 Gunakan resolusi untuk membuktikan kalimat ¬A∧¬B dari klausa dalam Latihan 7.20.

7.13 Latihan ini **menyelidiki** hubungan antara klausa dan kalimat implikasi.

a . Tunjukkan bahwa klausa ($\neg P_1 \vee \cdots \vee \neg P_m \vee Q$) secara logis setara dengan implikasinya kalimat ($P_1 \wedge \cdots \wedge P_m$) $\Rightarrow Q$.

b . Tunjukkan bahwa setiap klausa (terlepas dari jumlah literal positif) dapat ditulis dalam bentuk ($P_1 \wedge \cdots \wedge P_m$) $\Rightarrow (Q_1 \vee \cdots \vee Q_n)$, di mana Ps dan Qs adalah proposisi simbol. Basis pengetahuan yang terdiri dari kalimat-kalimat seperti itu dalam **implikasi normal bentuk** atau **bentuk Kowalski** (Kowalski, 1979).

IMPLIKATIF FORMULIR NORMAL

c . Tuliskan aturan resolusi penuh untuk kalimat dalam bentuk normal implikatif.

7.14 Menurut beberapa pakar politik, seseorang yang radikal (R) dapat dipilih (E) jika dia konservatif (C), tetapi sebaliknya tidak dapat dipilih.

a . Manakah dari berikut ini yang merupakan representasi yang benar dari pernyataan ini?

(i) $(R \wedge E) \Leftrightarrow C$
(ii) $R \Rightarrow (E \Leftrightarrow C)$
(iii) $R \Rightarrow ((C \Rightarrow E) \vee \neg E)$

b . Manakah dari kalimat dalam (a) yang dapat dinyatakan dalam bentuk Tanduk?

7.15 Pertanyaan ini dianggap mewakili masalah satisfiability (SAT) sebagai CSP.

a . Gambarkan grafik kendala yang terkait dengan masalah SAT

$$(\neg X_1 \vee X_2) \wedge (\neg X_2 \vee X_3) \wedge \ldots \wedge (\neg X_{n-1} \vee X_n)$$

untuk kasus khusus n = 5.

b . Berapa banyak solusi yang ada untuk masalah SAT umum ini sebagai fungsi dari n?

c . Misalkan kita menerapkan B ACKTRACKING -S EARCH (halaman 215) untuk menemukan *semua* solusi untuk SAT CSP dari tipe yang diberikan dalam (a). (Untuk menemukan *semua* solusi CSP, kami cukup memodifikasi algoritma dasar sehingga terus mencari setelah setiap solusi ditemukan.) Asumsikan itu variabel dipesan $X_1, \ldots, X_n$ dan false dipesan sebelum true. Berapa lama akankah algoritma akan berhenti? (Tulis ekspresi $O(\cdot)$ sebagai fungsi dari n.)

d . Kita tahu bahwa masalah SAT dalam bentuk Horn dapat diselesaikan secara linier dengan maju chaining (unit propagation). Kita juga tahu bahwa setiap CSP biner terstruktur dengan pohon diskrit, domain terbatas dapat diselesaikan dalam waktu linier dalam jumlah variabel (Bagian tion 6.5). Apakah kedua fakta ini terhubung? Bahas.

7.16 Jelaskan mengapa setiap klausa proposisional kosong, dengan sendirinya, memuaskan. Buktikan rig- dengan senang hati bahwa setiap set dari lima klausa 3-SAT adalah memuaskan, asalkan setiap klausa menyebutkan tepatnya tiga variabel berbeda. Apa set terkecil dari klausa seperti itu yang tidak memuaskan? Bangun set seperti itu.

7.17 Ekspresi *2-CNF* proposisional adalah gabungan dari klausa, masing-masing mengandung *tepat 2* liter, misalnya,

$$(A \vee B) \wedge (\neg A \vee C) \wedge (\neg B \vee D) \wedge (\neg C \vee G) \wedge (\neg D \vee G).$$

a . Buktikan menggunakan resolusi yang mencakup kalimat di atas G.

b . Dua klausa *berbeda secara semantik* jika tidak secara logis setara. Berapa banyak klausa 2-CNF yang berbeda secara semantik dapat dibangun dari n simbol proposisi?

c . Dengan menggunakan jawaban Anda untuk (b), buktikan bahwa resolusi proposisional selalu berakhir tepat waktu polinomial dalam n diberi kalimat 2-CNF yang berisi tidak lebih dari n simbol yang berbeda.

d . Jelaskan mengapa argumen Anda dalam (c) tidak berlaku untuk 3-CNF.

7.18 Pertimbangkan kalimat berikut:

[(Makanan ⇒ Pesta) ∨ (Minuman ⇒ Pesta)] ⇒ [(Makanan ∧ Minuman) ⇒ Pesta].

a . Tentukan, menggunakan enumerasi, apakah kalimat ini valid, memuaskan (tetapi tidak valid), atau tidak memuaskan.

b . Konversi sisi kiri dan kanan implikasi utama menjadi CNF, menunjukkan setiap langkah, dan jelaskan bagaimana hasilnya mengkonfirmasi jawaban Anda untuk (a).

c . Buktikan jawaban Anda untuk (a) menggunakan resolusi.

YG MEMISAHKAN FORMULIR NORMAL

7.19 Sebuah kalimat dalam **bentuk normal disjungtif** (DNF) jika merupakan disjungsi dari konjungsi dari literal. Misalnya, kalimat (A ∧ B ∧ ¬C) ∨ (¬A ∧ C) ∨ (B ∧ ¬C) ada dalam DNF.

a . Kalimat logika proposisional apa pun secara logis setara dengan pernyataan bahwa beberapa dunia yang benar di mana itu akan menjadi kenyataan sebenarnya terjadi. Dari pengamatan ini, buktikan bahwa kalimat apa pun dapat ditulis dalam DNF.

b . Bangun algoritma yang mengubah kalimat apa pun dalam logika proposisional menjadi DNF. (*Petunjuk* : Algoritma ini mirip dengan algoritma untuk konversi ke CNF yang diberikan dalam Bagian tion 7.5.2.)

c . Bangun algoritma sederhana yang mengambil sebagai input kalimat dalam DNF dan mengembalikan satis- tugas fying jika ada, atau melaporkan bahwa tidak ada tugas yang memuaskan.

d . Terapkan algoritme dalam (b) dan (c) ke serangkaian kalimat berikut:

A ⇒ B
B ⇒ C
C ⇒ ¬A.

e . Karena algoritma dalam (b) sangat mirip dengan algoritma untuk konversi ke CNF, dan karena algoritma dalam (c) jauh lebih sederhana daripada algoritma apa pun untuk menyelesaikan satu set sensor di CNF, mengapa teknik ini tidak digunakan dalam penalaran otomatis?

7.20 Konversi set kalimat berikut ke bentuk klausa.

S1: A ⇔ (B ∨ E).
S2: E ⇒ D.
S3: C ∧ F ⇒ ¬B.
S4: E ⇒ B.
S5: B ⇒ F.
S6: B ⇒ C

Berikan jejak pelaksanaan DPLL pada konjungsi dari klausa ini.

Halaman 51

7.21 Merupakan kalimat 4-CNF yang dibuat secara acak dengan n simbol dan m klausa lebih atau kurang cenderung dipecahkan daripada kalimat 3-CNF yang dibuat secara acak dengan n simbol dan m klausa? Menjelaskan.

7.22 Minesweeper, permainan komputer yang terkenal, terkait erat dengan dunia wumpus. Dunia kapal penyapu ranjau adalah kisi persegi N kotak dengan M ranjau tak terlihat tersebar diantara mereka. Setiap kotak dapat diperiksa oleh agen; kematian instan mengikuti jika tambang adalah diperiksa. Minesweeper menunjukkan keberadaan ranjau dengan mengungkapkan, di setiap kotak yang diperiksa, yang *jumlah* tambang yang secara langsung atau diagonal yang berdekatan. Tujuannya adalah untuk menyelidiki setiap kotak unmined.

a . Biarkan $X_{i,j}$ menjadi true iff kuadrat [i, j] berisi tambang. Tuliskan pernyataan yang persis seperti itu dua ranjau berbatasan dengan [1,1] sebagai kalimat yang melibatkan beberapa kombinasi logis dari ranjau $X_{i,j}$ proposisi.

b . Generalisasi pernyataan Anda dari (a) dengan menjelaskan cara membuat kalimat CNF menyatakan bahwa k dari n tetangga mengandung ranjau.

c . Jelaskan dengan tepat bagaimana agen dapat menggunakan DPLL untuk membuktikan bahwa persegi yang diberikan tidak (atau tidak) mengandung suatu tambang, mengabaikan batasan global bahwa ada persisnya tambang M

semuanya.

d . Misalkan kendala global dibangun dari metode Anda dari bagian (b). Bagaimana apakah jumlah klausa bergantung pada M dan N? Sarankan cara untuk memodifikasi DPLL bahwa kendala global tidak perlu diwakili secara eksplisit.

e . Apakah ada kesimpulan yang diperoleh dengan metode di bagian (c) batal ketika global kendala diperhitungkan?

f . Berikan contoh konfigurasi nilai probe yang menginduksi *dependensi jarak jauh* sedemikian rupa sehingga isi dari kotak yang tidak disediakan akan memberikan informasi tentang isi dari kotak yang jauh. (*Petunjuk* : pertimbangkan papan N × 1.)

7.23 Berapa lama untuk membuktikan KB | = α menggunakan DPLL ketika α *sudah* literal *terkandung dalam* KB? Menjelaskan.

7.24 Lacak perilaku DPLL pada basis pengetahuan pada Gambar 7.16 saat mencoba buktikan Q, dan bandingkan perilaku ini dengan algoritma forward-chaining.

7.25 Tuliskan aksioma status penerus untuk predikat Terkunci, yang berlaku untuk pintu, seperti jumlah tindakan yang tersedia adalah Kunci dan Buka Kunci.

7.26 Bagian 7.7.1 memberikan beberapa aksioma negara penerus yang diperlukan untuk wumpus dunia. Tulis aksioma untuk semua simbol fasih yang tersisa.

7.27 Memodifikasi H YBRID -W UMPUS -A GENT menggunakan 1-CNF logis estimasi state metode yang dijelaskan pada halaman 271. Kami mencatat pada halaman itu bahwa agen seperti itu tidak akan dapat untuk memperoleh, mempertahankan, dan menggunakan keyakinan yang lebih kompleks seperti disjungsi $P_{3,1} \vee P_{2,2}$. Sanj menggunakan metode untuk mengatasi masalah ini dengan mendefinisikan simbol proposisi tambahan, dan coba saja di dunia wumpus. Apakah itu meningkatkan kinerja agen?