1 A kockavilág

1.1 Problématér

Egy asztalon kockák állnak. A kockákat egymásra lehet helyezni. Egy kockára csak egy másikat lehet helyezni és minden kocka csakis egy másik kockán helyezkedhet el.

A problématérben állításokat fogalmazhatunk meg kockákról és változókról, melyek tetszőleges kockát jelölhetnek, ezeket közösen elemeknek nevezzük. A kockákat különböző objektumoknak tekintjük.

Fontos kitétel, hogy a kockavilág zárt, előre meg van adva, hogy mely kockák használhatóak fel egy feladat megoldására.

A kockavilág monoton, olyan értelemben, hogy bármely ellentmondást nem tartalmazó feladat megoldható elegendő kockával. Ennek az az oka, hogy a kockavilágban nem fejezhető ki olyan tulajdonság, hogy például "csak 3 torony lehetséges".

1.2 Atomi állítások

- Az X elem az asztalon áll: $\underline{X}$
- Az X elemen nincs másik kocka: $\overline{X}$
- Az X elem rajta van a Y elemen: $\frac{X}{Y}$
- Az X és Y elemek megegyeznek: $X = Y$
- Illetve ezek tagadása: $\neg\,\underline{X}$, $\neg\,\overline{X}$, $\neg\frac{X}{Y}$, $X \neq Y$

1.3 Feladat

A feladat, hogy bizonyítsunk egy állítást a kockavilágon belül. Ez egy $C_1 \wedge C_2 \wedge \cdots \wedge C_n \Rightarrow R$ alakú állítás, ahol C_i, R termek. Erről kell eldöntenünk, hogy igaz-e. Kiegészítő cél lehet az, hogy amennyiben ez nem teljesül, adjunk ellenpéldát.

1.4 Példaproblémák

1.4.1 Egyszerűbb példák

Az egyszerűbb példákat könnyű állapottér kereséssel megoldani, de a rezolúció esetén a következtetés elakadhat, ha nincs megfelelő szabály.

- $I = \{A, B\} : \overline{A} \wedge \underline{B} \wedge \frac{X}{Y} \Rightarrow \frac{A}{B}$
- $I = \{A, B\} : \neg \overline{B} \wedge \neg \underline{A} \Rightarrow \frac{A}{B}$
- $I = \{A, B, C\} : \neg \overline{C} \wedge \neg \overline{B} \wedge \neg \underline{B} \wedge \neg \underline{A} \Rightarrow \frac{A}{B}$

1.4.2 Összetettebb példa

Egy már kicsit komplikáltabb, többváltozós probléma. Öt kockából kell egy háromszintes tornyot építeni és két egyszinteset.

$I = \{A, B, C, D, E\} : \overline{A} \wedge \neg \underline{A} \wedge \neg \overline{B} \wedge \neg \frac{B}{D} \wedge \underline{C}$
$\wedge \neg \frac{D}{B} \wedge \overline{E} \wedge \frac{X}{Y} \wedge \frac{Y}{Z} \wedge \underline{U} \wedge \underline{V} \wedge \underline{W} \Rightarrow \frac{A}{B} \wedge \frac{B}{C}$

1.4.3 A két torony problémacsalád

A két torony problémacsalád egy egyszerű sémán alapul: két tornyot kell építeni változókból, de nincs elegendő kocka a fölépítéséhez. Ennek következményeképp beláthatjuk, hogy a két torony megegyezik. Ezek jellemzően nehezen oldhatók meg állapottér kereséssel. A rezolúció új információkra bukkanhat, ha vannak beépítve olyan metaszabályok, mint hogy egy kocka egy másik kocka fölött van. Ha a rezolúció belátja, hogy a kockák diszjunktak, akkor tovább csökkenthetjük a keresési teret.

A két torony probléma n paramétere adja meg, hogy milyen magas tornyokat akarunk építeni.

$I = \{A_1, ..., A_{2n-1}\}$:
$\frac{X_n}{X_{n-1}} \wedge ... \wedge \frac{X_2}{X_1} \wedge \underline{X_1} \wedge \frac{Y_n}{Y_{n-1}} \wedge ... \wedge \frac{Y_2}{Y_1} \wedge \underline{Y_1} \Rightarrow X_1 = X_2$

2 Reprezentáció

A reprezentációt bemutató kódrészletek haskell programozási nyelvben készültek.

2.1 Állítások reprezentációja

Az állításokat célszerű valamilyen normálformában reprezentálni, mivel így a logikai következtetések egyszerűbbek, és könnyen kinyerhető az egyes elemekre vonatkozó információ. Ezt a reprezentációt azonban nem csak a következtetések, hanem a modell keresés is használja.

```
data CNF = CNF [Term]
data Term = Term [Atom]
data Atom = Prop Proposition | Not Proposition
data Proposition = Elem `On` Elem | Top Elem | Bottom Elem
                 | Elem :=: Elem
data Elem = Final String | Var String
```

2.2 Modell reprezentációja

A modell fontos tulajdonsága, hogy a kockákból épített oszlopok sorrendje nem számít, erre vonatkozóan nem tudunk állításokat megfogalmazni.

Így a modell reprezentálható sorozatok halmazával, melyekre igaz, hogy minden elem pontosan egy sorozatban szerepel. A modell kockáiba belehelyezzük azokat a megkötéseket is, amik az adott kockára vonatkoznak, így ezeket lokálisan is elérjük. Amikor tudjuk, hogy két kocka egymáson van, akkor azokat egy blokknak tekintjük.

```
data BlockConstraint = MustBe | CanBe | CantBe
data Block = Block { items :: [String]
                   , isTop :: BlockConstraint
                   , isBottom :: BlockConstraint
                   }
type Model = Set [Block]
```

2.3 Segédreprezentációk

A bizonyításokhoz segítséget tud nyújtani, ha kiegészítő információkat is végigvezetünk a bizonyítás során.

2.3.1 Ekvivalenciatábla

Gyakran van szükségünk arra, hogy megvizsgáljuk, egy változó milyen értéket vehet föl. Ez az információ kiolvasható a CNF-ből is, ám gyorsabb, ha ezt az információt egy táblázatban is nyilvántartjuk. El tudjuk tárolni, hogy az

adott változó milyen kockáknak felelhet meg, illetve, hogy milyen változókkal eshet egybe.

Amennyiben egy változó már csak egy kockának felelhet meg, akkor egyszerűsíthetjük az állításainkat, átírva benne a változót az adott kockára.

```
type PossibleValues = ([String],[String])
type EqTable = Map String PossibleValues
```

2.3.2 Segédállítások

Ahhoz, hogy a következtetés lehetőségeit bővítsük, segédállításokat kell fölvennünk. Vegyük az egyszerű esetet, ahol a $\frac{X}{Y}$ és $\frac{Y}{Z}$ állítások teljesülnek. Ebben az esetben lehet egy szabályunk: $\frac{U}{V} \Rightarrow U{\neq}V$

Azonban ezzel hiába próbálkozunk, nem tudjuk az $X{\neq}Z$ állítást is bebizonyítani. Ehhez szükségünk van egy olyan állításra, amely kifejezi, hogy egy elem a másik fölött van. Ezt jelöljük az $\frac{\frac{U}{\vdots}}{V}$ kifejezéssel.

Ennek a levezetésére és felhasználására alkalmazzuk a következő szabályokat:

$$\frac{U}{V} \Rightarrow \frac{\frac{U}{\vdots}}{V}, \qquad \frac{U}{V} \wedge \frac{\frac{V}{\vdots}}{W} \Rightarrow \frac{\frac{U}{\vdots}}{W}, \qquad \frac{\frac{U}{\vdots}}{V} \Rightarrow U{\neq}V$$

3 Keresési módszerek fajtái

3.1 Logikai következtetés

Amikor logikai következtetéssel bizonyítjuk az állítást, az ismert állításokból új állításokat vezetünk le, amíg a keresett állítást be nem bizonyítjuk.

A logikai következtetés módszere lineáris, új ismereteket adunk a rendszerhez, amíg csak el nem érkezünk a keresett állításhoz. Jellemzően egy következtetés cáfolatát keressük. Ebben az esetben ha az eljárás elakad, mielőtt megcáfolhatnánk az állítást, akkor az állítás igaz, amennyiben a következtetési módszerünk teljes.

3.2 Modell keresés

A modell keresés során modelleket építünk, és megpróbálunk olyat keresni, ami ellentmond az állításnak. Amennyiben bebizonyítjuk, hogy egy ilyen modell sem található, akkor beláttuk az állítás igazságát.

A modell keresés implementációban egy visszalépéses keresésként vagy gráfkeresésként valósítható meg a legkönnyebben. A módszer nehézségét a keresési tér nagysága jelenti.

3.3 Hibrid módszerek

Hibrid módszernek nevezzük a következtetés és a keresés különféle kombinációit. Ezekre jellemző, hogy a modell-kereséshez hasonlóan több lehetséges változatát vizsgálják meg a megoldásnak.

4 Egyszerű rezolúciós módszer és korlátai

Az egyszerű rezolúciós módszer egy olyan eljárás, amellyel az állítások halmaza bővíthető. Azonban ez a módszer nem teljes, nem minden állítás bizonyítható benne.

A módszer konjunktív normálformán ($t_1 \wedge t_2 \wedge \cdots \wedge t_n$, ahol $t_i = a_{i,1} \vee a_{i,2} \vee \cdots \vee a_{i,n_i}$, melyben $a_{i,j}$ egy atomi állítás vagy annak tagadása) dolgozik, egy ekvivalenciatábla segítségével. A módszer az ekvivalencia-táblát is karbantartja, azok az állítások, amik az ekvivalencia-táblára vannak hatással átkerülnek oda a rezolúciós lépés után. Amennyiben az ekvivalencia-táblából kiolvasható egy változó értéke, akkor egy átírási lépés során ezt érvényesítjük.

A módszer alapja egy rezolúciós lépés, mely során a $a_1 \vee \cdots \vee a_i \vee c \vee a_{i+2} \vee \cdots \vee a_m$ és egy $b_1 \vee \cdots \vee b_j \vee d \vee b_{j+2} \vee \cdots \vee b_n$ alakú termből indulunk ki. Amennyiben be tudjuk látni, hogy c és d kizárják egymást, akkor ezzel bizonyítjuk az $a_1 \vee \cdots \vee a_i \vee a_{i+2} \vee \cdots \vee a_m \vee b_1 \vee \cdots \vee b_j \vee b_{j+2} \vee \cdots \vee b_n$ állítást.

Fontos eldönteni, hogy milyen esetekben tekintünk két atomot ellentmondónak. A módszer kidolgozásakor az alábbi állításokat soroltuk ide:

- Egy állítás és annak logikai tagadása.
- $\dfrac{X}{Y}$ és $\dfrac{X}{}$
- $\dfrac{X}{Y}$ és $\dfrac{}{Y}$

- $\frac{X}{A}$ és $\frac{X}{B}$
- $\frac{A}{X}$ és $\frac{B}{X}$

Itt az X és Y jelölhet változót és kockát is, az A és B kizárólag kockát (mert ezekről tudjuk, hogy nem egyezhetnek meg).

Az egyszerű rezolúciós módszer, bár kevés esetben találja meg a megoldást, gyorsan működik, nem okozza az állítások gyors elszaporodását.

5 Állapottér keresés és korlátai

Az állapottér keresés során megpróbálunk egy olyan modellt építeni, ami megcáfolja a hipotézisünket. Amennyiben nem sikerül ilyet építeni, akkor bizonyítottnak tekintjük a hipotézist. Ebből látszik, hogy az állapottér keresés egy teljes módszer, minden lehetséges modellt végigpróbál, vagy próba nélkül belátja, hogy nem elégítheti ki a feltételeket.

Vágások nélkül az állapottér igen nagy, n különböző kocka lehetséges elrendezéseinek száma $n!$ és $(n+1)!$ közötti. (Ha a kockákat kötelezően egy oszlopba rendezzük, akkor tetszőleges sorrendben állhatnak, így $n!$ számú állapot lehetséges. A felső korlát empirikus adatok alapján igazolódni látszik.[1])

Vágási lehetőség akkor van, amikor az állítások alapján a keresési fa egy egész ágát kihagyhatjuk. Például, ha tudjuk, hogy $\neg\frac{A}{B}$ és a jelenlegi lépés során az A kocka a B-re kerülne, akkor az egész ágat kihagyhatjuk, ami a jelenlegi lépésből eredne.

A modell ellenőrzésének több lépése lehet:

1. Azokat a feltételeket, melyeket berakás közben is ellenőrizni tudunk, azonnal ellenőrizzük.

2. Azokat a feltételeket, amelyeket berakás során nem tudunk rögtön ellenőrizni, de amikben változók nem szerepelnek a modell befejezése után ellenőrizhetjük le.

[1] A lehetséges modellek száma $n = 1..20$ kocka esetén: 1, 3, 9, 37, 169, 981, 6429, 49669, 430861, 4208925, 45345165, 536229373, 6884917597, 954730 49469, 1420609412637, 22580588347741, 381713065286173, 6837950790434781, 129378941557 961565, 2578133190722896861. Az $(n+1)!$ értéke végig a sorozat fölött marad és a két sorozat hányadosa 0-hoz tart.

3. Azokat a feltételeket, melyekben változók is szerepelnek, csak azután tudjuk ellenőrizni, hogy a változókhoz értékeket rendelünk. Az állapottér keresés gyengesége, hogy ez nagyon költségessé teheti az eredmények ellenőrzését. Ha van n kockánk és k változónk, akkor k^n a változók lehetséges értékeinek (változó-interpretációk) száma. Meg kell vizsgálni, hogy van-e legalább egy olyan interpretáció, amiben minden állítás ki lesz elégítve. Amennyiben van ilyen, akkor a modell helyes.

Arra, hogy milyen módszerrel építjük fel a modelleket, két módszert találtam:

5.1 Fix sorrendű elhelyezés

Vegyük a kockákat eredeti sorrendben. Egyenként helyezzük el őket a modellben úgy, hogy új oszlopba tesszük őket, vagy egy már létező oszlopban tetszőleges helyre beszúrjuk.

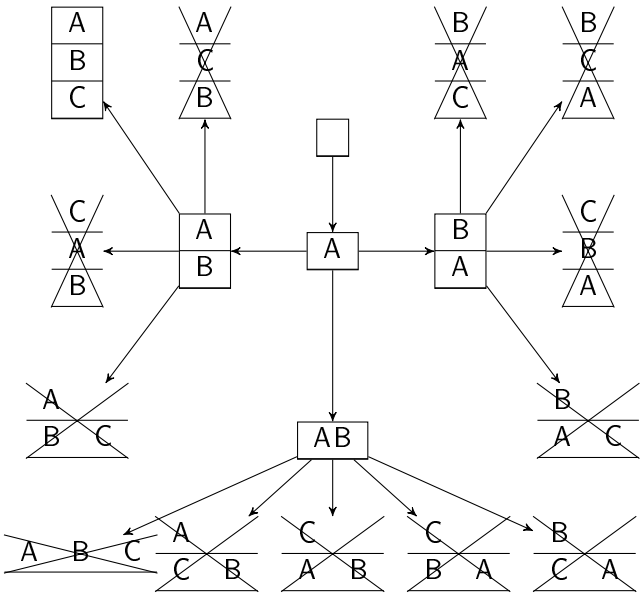
Ennek a módszernek az előnye az, hogy nem vizsgáljuk meg ugyanazt a modellt kétszer, ez a módszer minden modellt csak egyszer talál meg.

Lehetséges optimalizálások:

- Ha tudjuk hogy $\frac{A}{B}$, akkor a kettőt egy AB kockaként kezeljük.
- Az alsó kockákat előzőleg elhelyezzük külön oszlopokba, ezek nem fogják a keresési teret növelni.
- A felső kockákat az összes többi kocka elhelyezése után rakjuk rá az oszlopok tetejére, így véletlenül sem tudunk rájuk rakni más kockákat.

5.1.1 Példa

Az állapottér keresés a háromkockás feladatra fix sorrendű elhelyezésben.



Látható, hogy az összes megoldást végig kellett néznünk, mert csak a kész modelleket tudtud megcáfolni. Ennek az az oka, hogy a fix sorrendű elhelyezésben nem tudjuk fölhasználni a negatív állításokat.

5.2 Fix elhelyezkedésű elrendezés

A második módszer során vegyük a kockák permutációit és ennek megfelelő sorrendben helyezzük el őket. Ehhez persze nem kell az összes permutációt előre előállítani, elég ha minden lépésben veszünk egy tetszőleges, még nem használt kockát és elhelyezzük. Kezdhetünk új oszlopot, vagy valamelyik már lerakott oszlop tetejére rakhatunk egy kockát.

Ennek a módszernek az előnye, hogy egy kocka elhelyezése során több információt szerzünk. Ha például A kockát B-re helyeztük, akkor biztos, hogy $\frac{A}{B}$ teljesülni fog, hiszen nem szúrhatunk később a kettő közé egy újabb kockát.

Amennyiben ezt a módszert használjuk, gondoskodnunk kell arról, hogy a többször megtalált állapotokat ne terjesszük többször ki. Ilyen előfordul már két kocka esetén is. Lehetséges modellek az A és B kockákkal: $\frac{\mathrm{B}}{\mathrm{A}}$, $\underline{\mathrm{AB}}$, $\frac{\mathrm{A}}{\mathrm{B}}$, $\underline{\mathrm{BA}}$, A második és a negyedik modell ekvivalens.

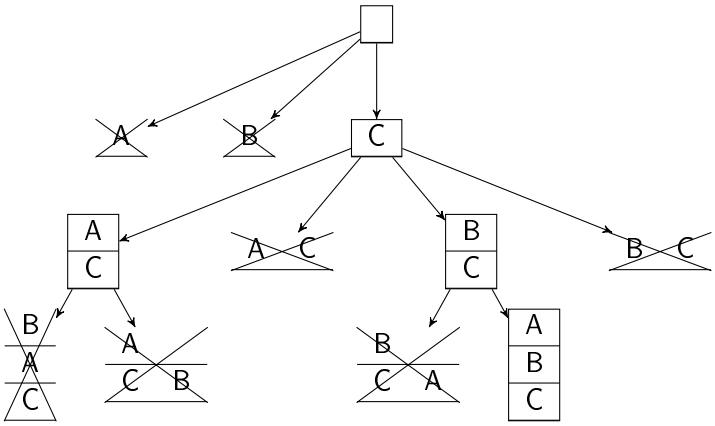
Erre a problémára egy lehetséges megoldás, ha ekvivalenciát definiálunk a modellek között. A redundancia miatt ezzel a módszerrel nagyobb állapotteret kell földeríteni, mint a fix sorrendű elhelyezéssel, ami legalábbis adminisztrációs költséget jelent, ha ekvivalenciákat vizsgálunk. Emiatt csak akkor érdemes alkalmazni, ha ki tudjuk használni az így nyert plusz információt.

A módszer segítségével a következő optimalizációk valósíthatók meg a fix sorrendű elhelyezés optimalizációi mellett:

- Ha tudjuk hogy $\neg\frac{A}{B}$, akkor az A blokkot nem helyezhetjük el a B blokkon.
- Ha tudjuk, hogy $\neg\ \overline{A}$, akkor A lerakása után a következő kockát rögtön A-ra rakhatjuk.
- Ha tudjuk, hogy $\neg\ \underline{A}$, akkor A-t csak már meglevő torony tetejére rakjuk, új toronyba nem.

5.2.1 Példa

Az állapottér keresés a háromkockás feladatra fix elhelyezkedésű elrendezésben. Az előző példával szemben itt ki tudjuk használni, hogy előzetes vágásokat tudunk tenni.



Láthatjuk, hogy ebben az esetben sok lehetséges állapotot hamar ki tudunk szűrni, mivel a negatív állításokat is hasznosítani tudjuk.

6 Technikák a keresési tér csökkentésére

6.1 Anonim kockák

Bizonyos feladatokban valamely kockák egyáltalán nem jelennek meg. Ilyen például a kéttorony-probléma is, ahol egy konkrét kocka sem jelenik meg az állítások között, csak változók (a 2. oldalon található). Az anonim kockáknak csak a száma befolyásolja a bizonyítást, mert ez dönti el, hogy valamely feladathoz van-e elég kockánk, de annak, hogy milyen sorrendben használjuk fel őket, nincs jelentősége.

Állapottér keresések esetén a keresése teret hatalmasra növeli az, hogy az egyébként egymással teljes mértékben felcserélhető kockák sorrendjeit is külön kezeljük.

6.2 Redundáns változók kiküszöbölése

Azok a változók, amik csak egy $\frac{X}{Y}$ állításban szerepelnek elhagyhatók, az adott szabály pedig más alakra írható. Az $\frac{X}{Y}$ változó átírható $\neg \underline{\quad X \quad}$-re, ha Y nem szerepel máshol, és $\neg \overline{\quad Y \quad}$-ra, ha X nem szerepel máshol.

7 Hibrid módszer

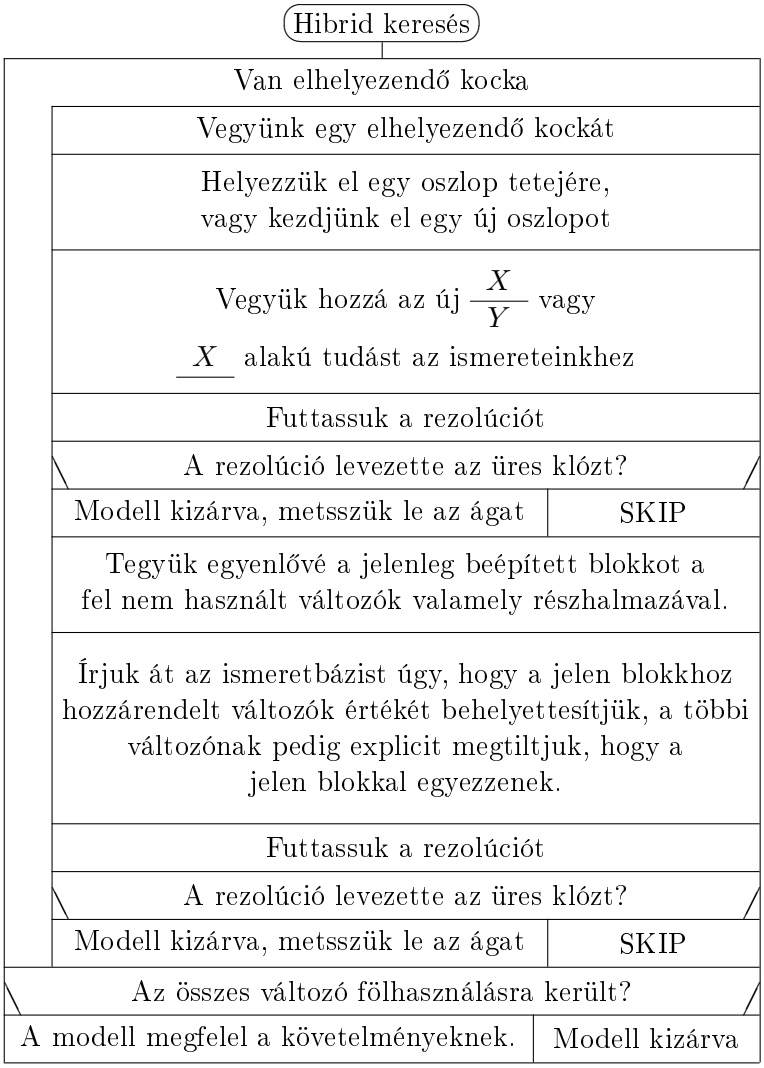
Az általam megtervezett hibrid módszer az egyszerű rezolúciós algoritmust használja arra, hogy az állapottér-keresés terét lecsökkentse. Az algoritmus kidolgozásakor fontos cél volt, hogy a lehetséges modell és az interpretáció keresése egyszerre történjen, így ezek egymásnak kölcsönösen új információkat biztosíthatnak, ami következtében a keresés gyorsabb és stabilabb lesz.

Ahhoz, hogy nagyobb mennyiségű információt nyerjünk ki, fix elhelyezkedésű elrendezést használjunk. Minden egyes lépésben, amellett, hogy elhelyezünk egy kockát, megnézzük, hogy az elhelyezett kocka mely változókkal egyezhet meg. Mindkét lépés során rezolúciós lépésekkel megpróbálunk ellentmondást keresni, hogy levágjuk az adott keresési ágat.

Az egész keresés történhet visszalépéses kereséssel, vagy gráfkereséssel. Az előbbi esetben nem kell egy potenciálisan nagy halmazt fönntartani az összes bejárt állapottal, a másodikban viszont kizárjuk azt, hogy ugyanazokat az állapotokat többször is megtaláljunk a keresés során és így fölösleges

munkát végezzünk. Ennek megfelelően a metszés jelentheti azt, hogy visszalépünk, de azt is, hogy a csúcsot kiterjesztés nélkül a zárt halmazba helyezzük.

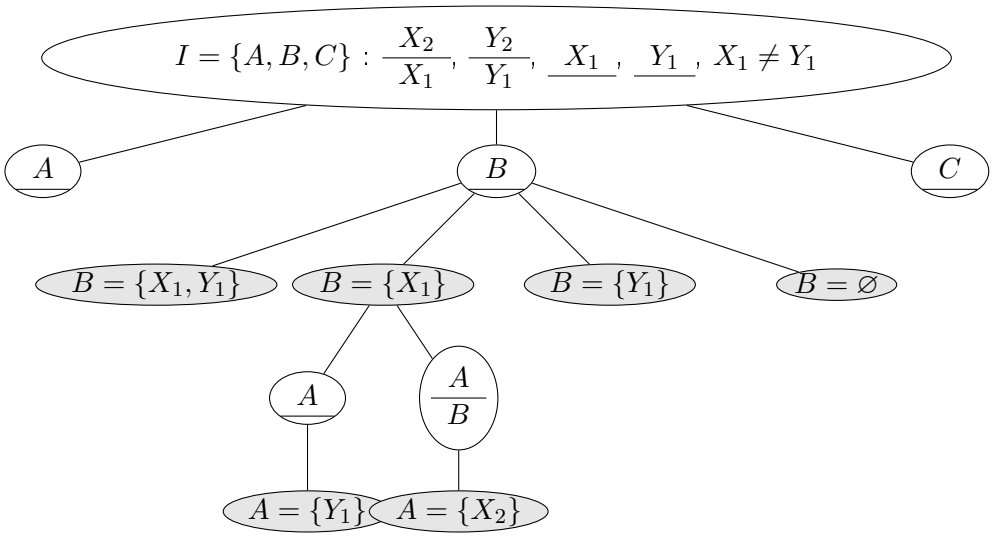
7.1 Algoritmus



7.2 Példa

A két torony probléma megoldás $n = 2$ esetén hibrid kereséssel, anonimizált blokkokkal.

A szürkén jelzett csomópontokban változók rögzítése történik, még a fehérekben a kockák elhelyezése. A példa nem tartalmazza az összes lehetséges állapotot, csak bemutatja a módszer működését. A gyermekcsomópontokban levő állítások hozzáadódnak a szülőcsomópontokban levő ismeretekhez. A gyökér tartalmazza az eredeti feladatot.



Ha ehhez a példákhoz még a kockák anonimizálását is felhasználjuk, akkor nem számít a kifejtések sorrendje. Így elsőként mindenképp a $\underline{A}$ csomópont lenne kifejtve.