

# Igényvezérelt heurisztikaszámítás informált kereséssel támogatott absztrakció alapú modellellenőrzésben

*Vörös Asztrik*

*Konzulens*

*Szekeres Dániel*



Budapest University of Technology and Economics  
Department of Measurement and Information Systems  
Critical Systems Research Group



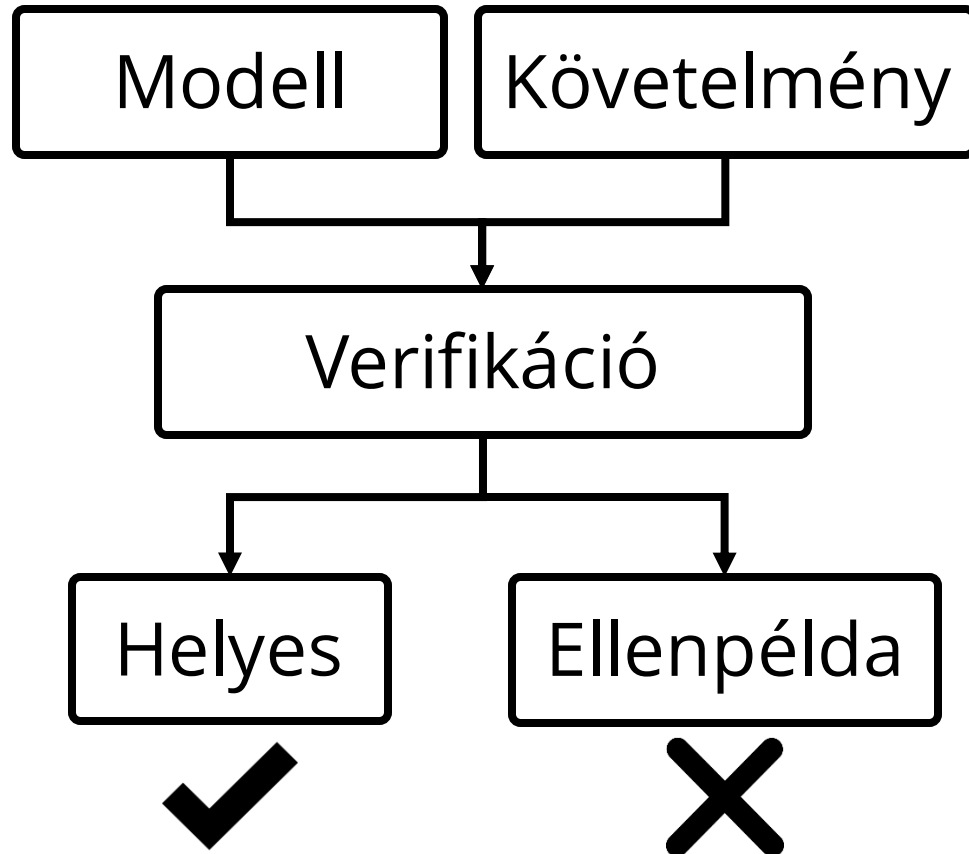
# Kritikus beágyazott rendszerek

- **Helyesség** biztosítása fontos
- Nem elég egyes viselkedések vizsgálata
- **Modellellenőrzés:** minden lehetséges viselkedés ellenőrzése

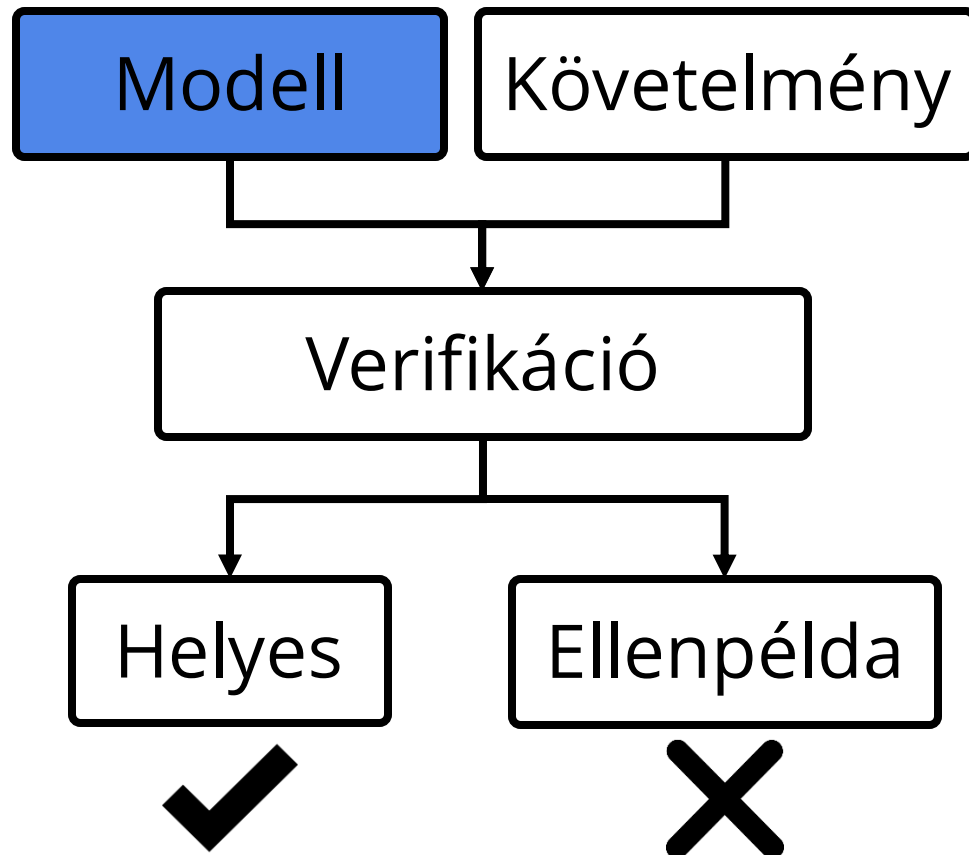




# Modellellenőrzés



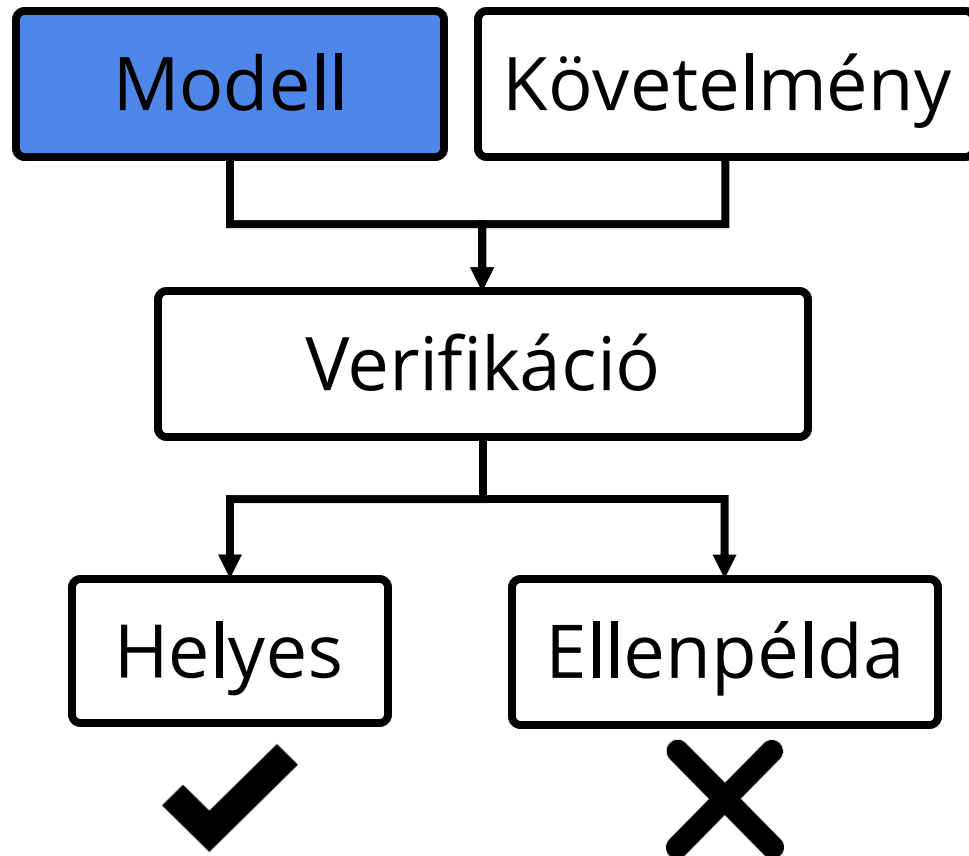
# Modellellenőrzés



## C nyelvű forráskód

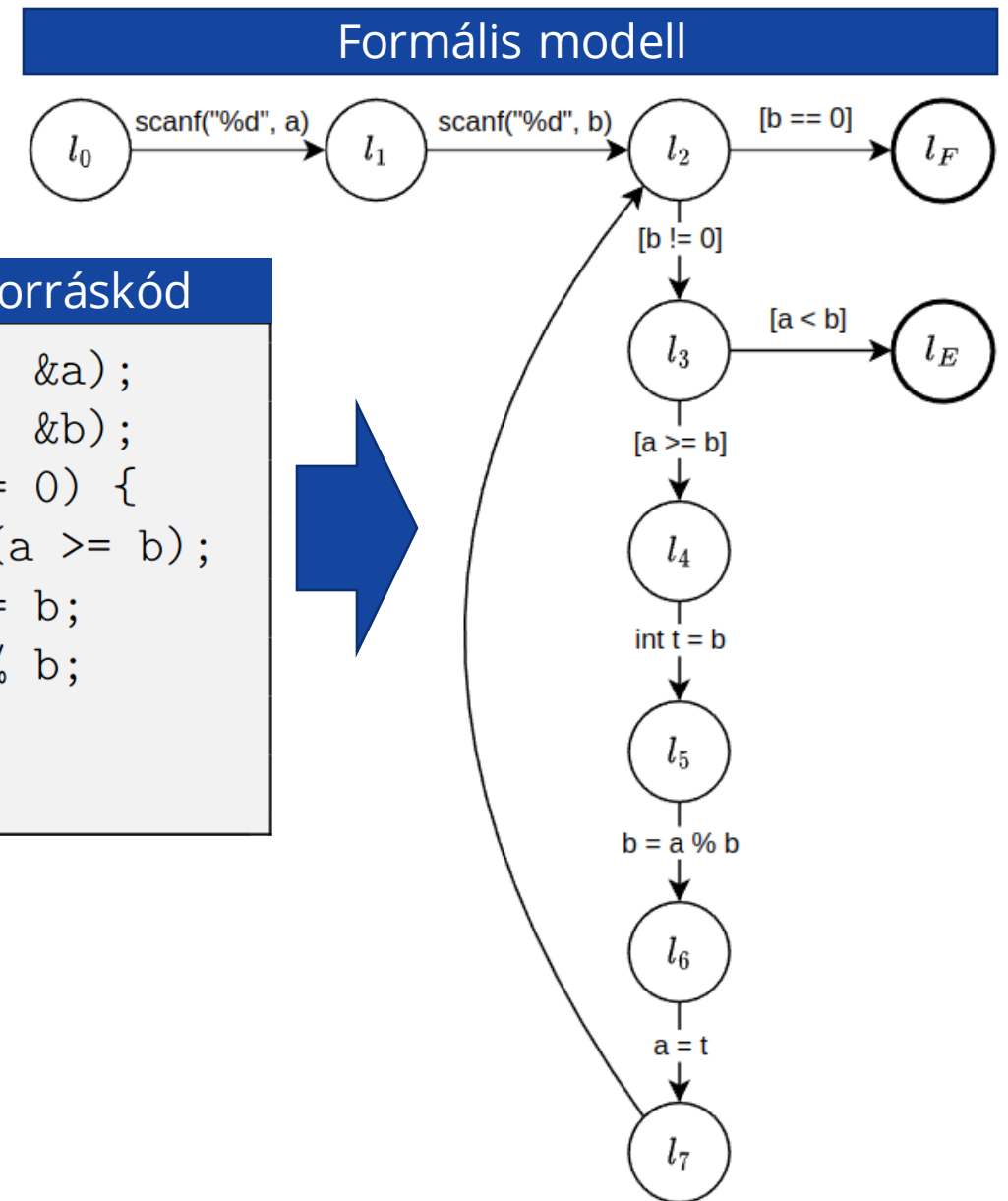
```
scanf("%d", &a);
scanf("%d", &b);
while (b != 0) {
    assert(a >= b);
    int t = b;
    b = a % b;
    a = t;
}
```

# Modellellenőrzés

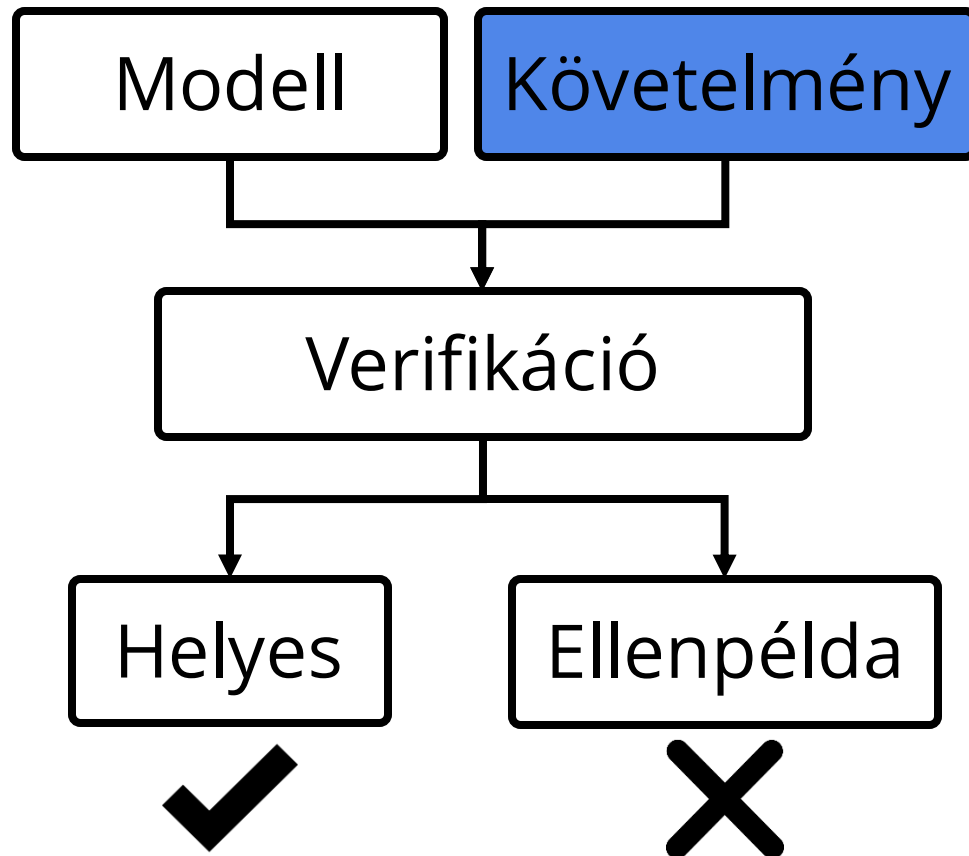


C nyelvű forráskód

```
scanf("%d", &a);
scanf("%d", &b);
while (b != 0) {
    assert(a >= b);
    int t = b;
    b = a % b;
    a = t;
}
```

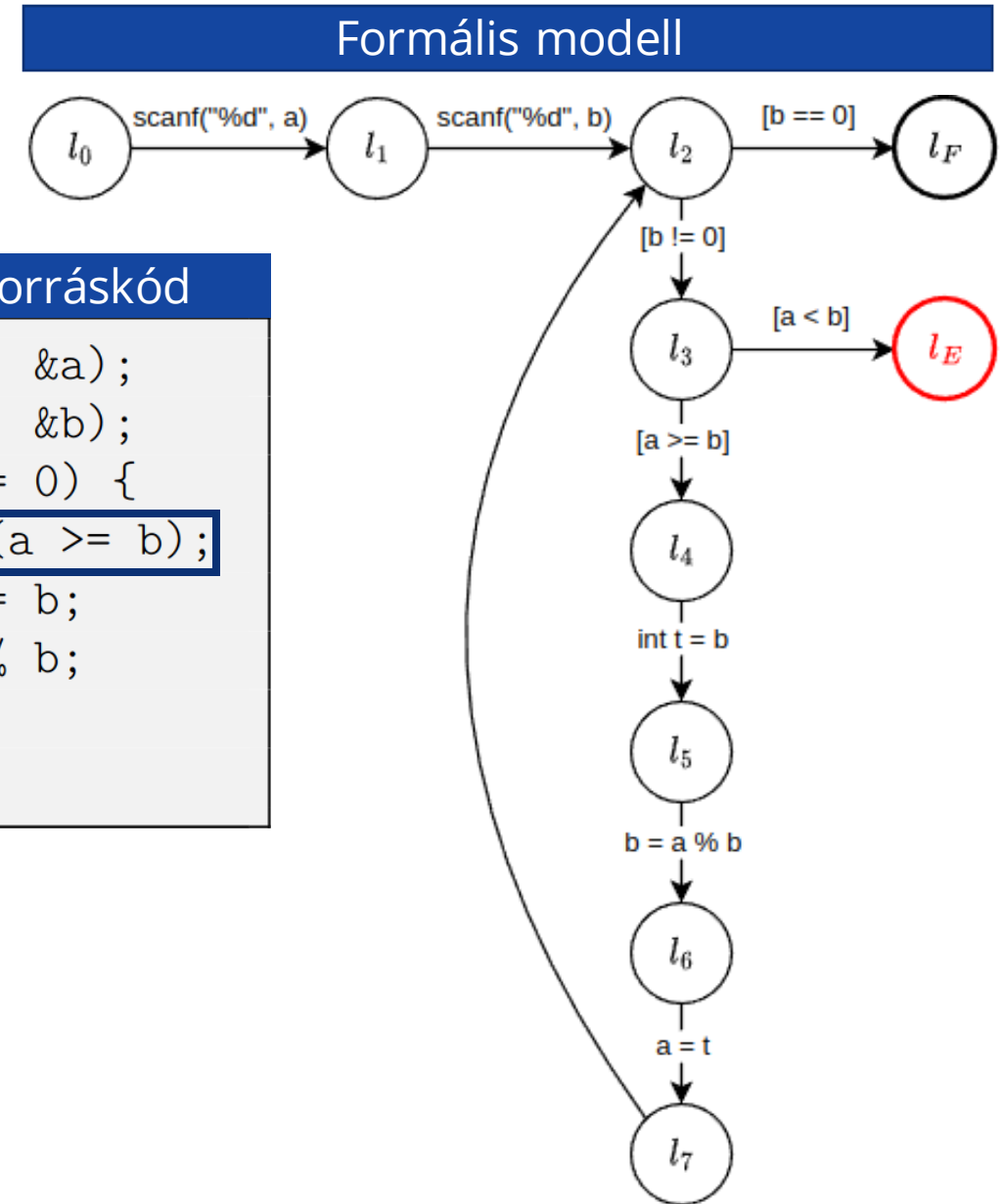


# Modellellenőrzés

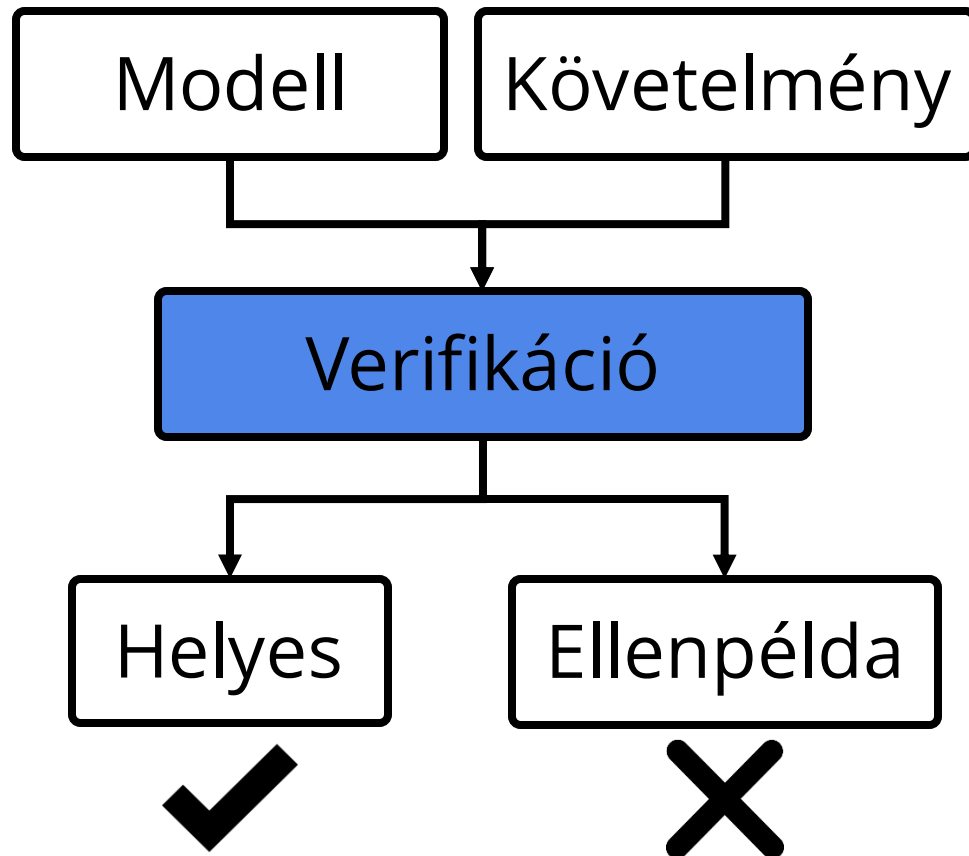


C nyelvű forráskód

```
scanf("%d", &a);
scanf("%d", &b);
while (b != 0) {
    assert(a >= b);
    int t = b;
    b = a % b;
    a = t;
}
```

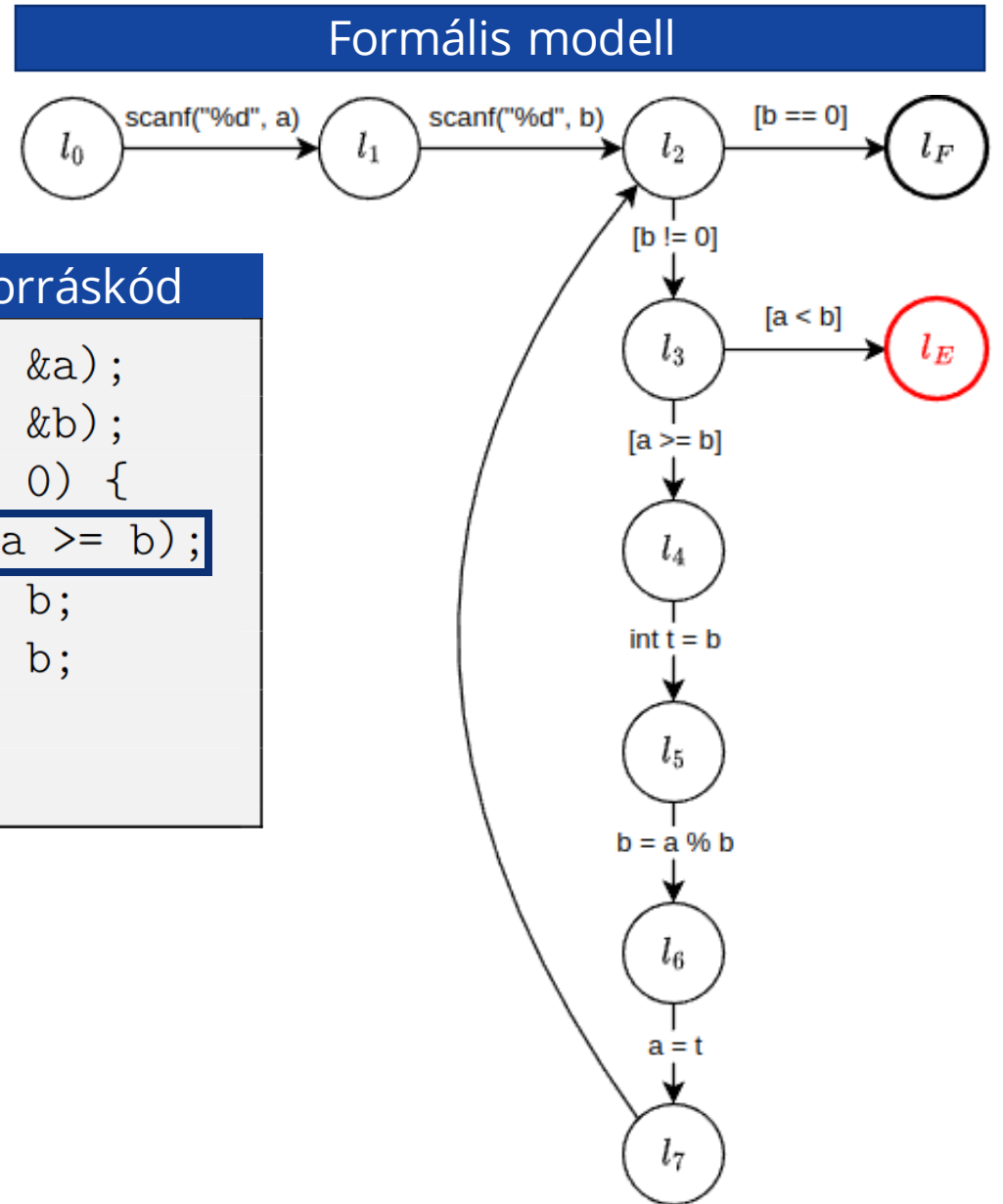


# Modellellenőrzés

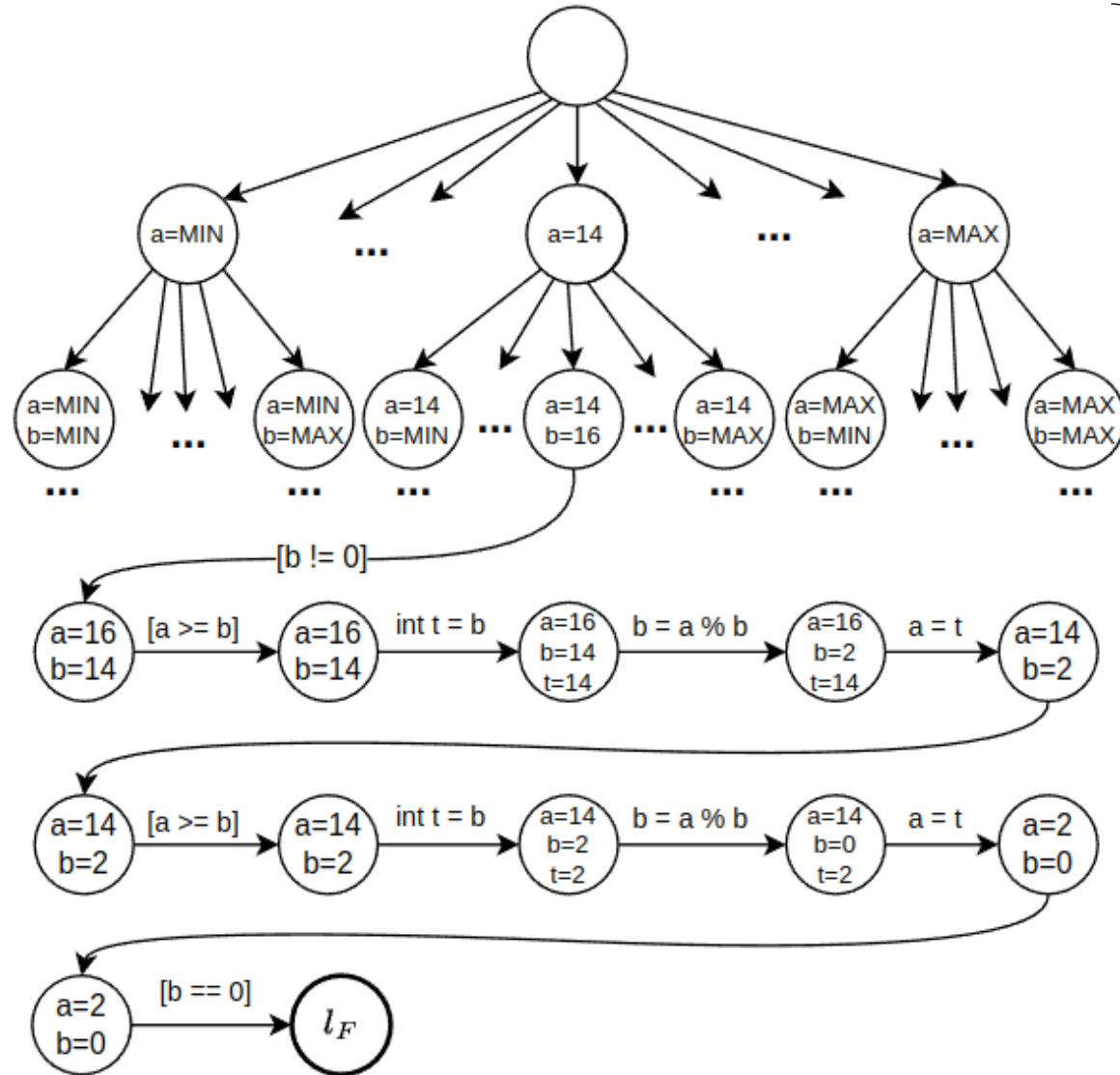


C nyelvű forráskód

```
scanf("%d", &a);
scanf("%d", &b);
while (b != 0) {
    assert(a >= b);
    int t = b;
    b = a % b;
    a = t;
}
```



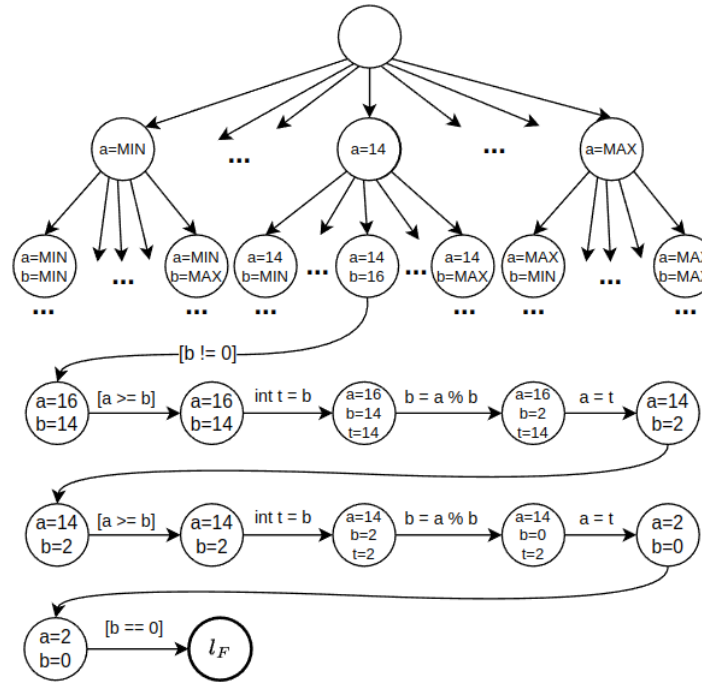
# Verifikáció: Állapottér bejárás



Hibás állapotok  
**keresése**



# Verifikáció: Állapottér bejárás

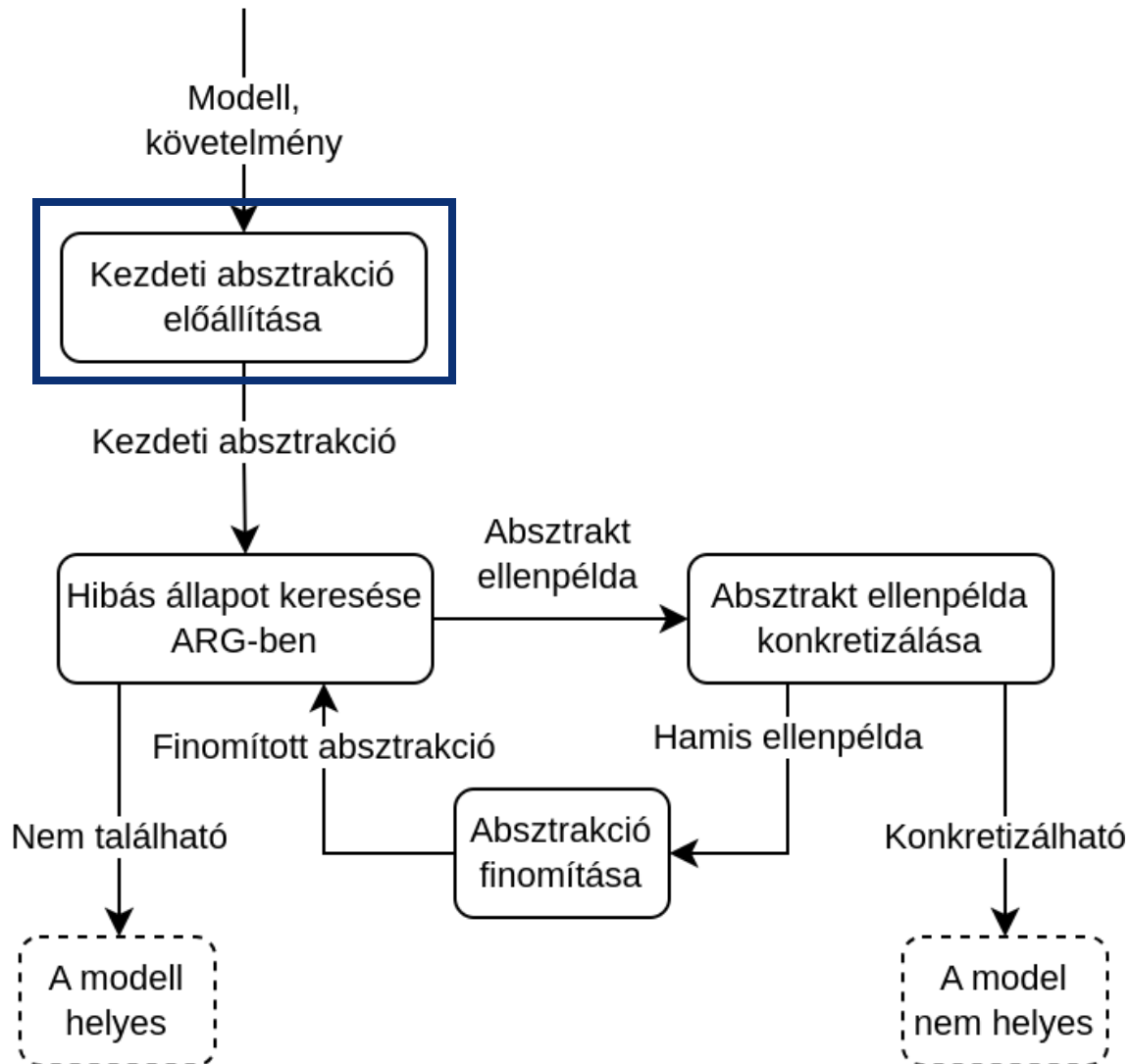


- **Állapottér-robbanás**

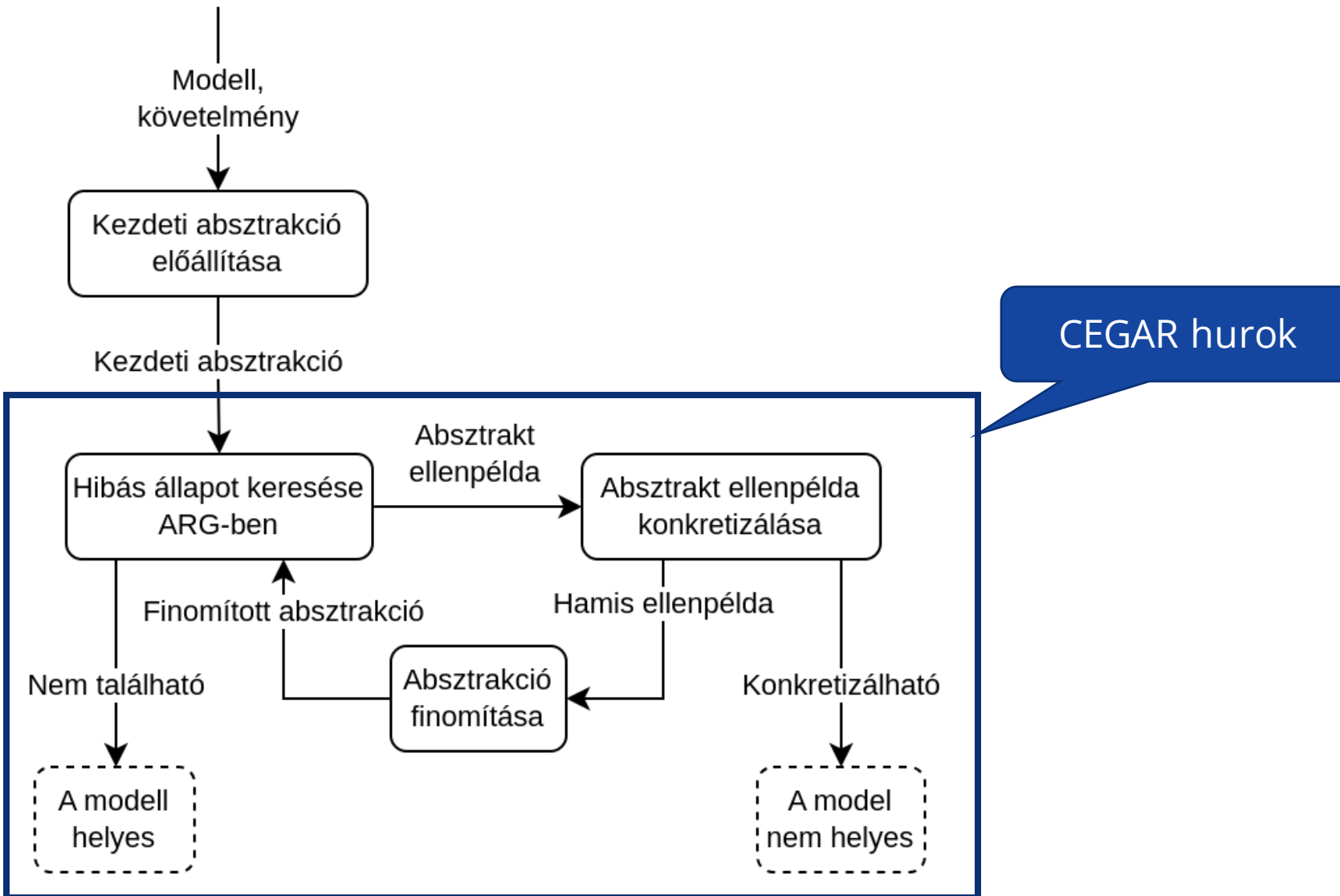
- Szoftverek adat változói sokféle kombinációban előállhatnak
- Állapottér mérete exponenciálisan nő a változók számában

- Megoldás: **absztrakció** alapú modellellenőrzés

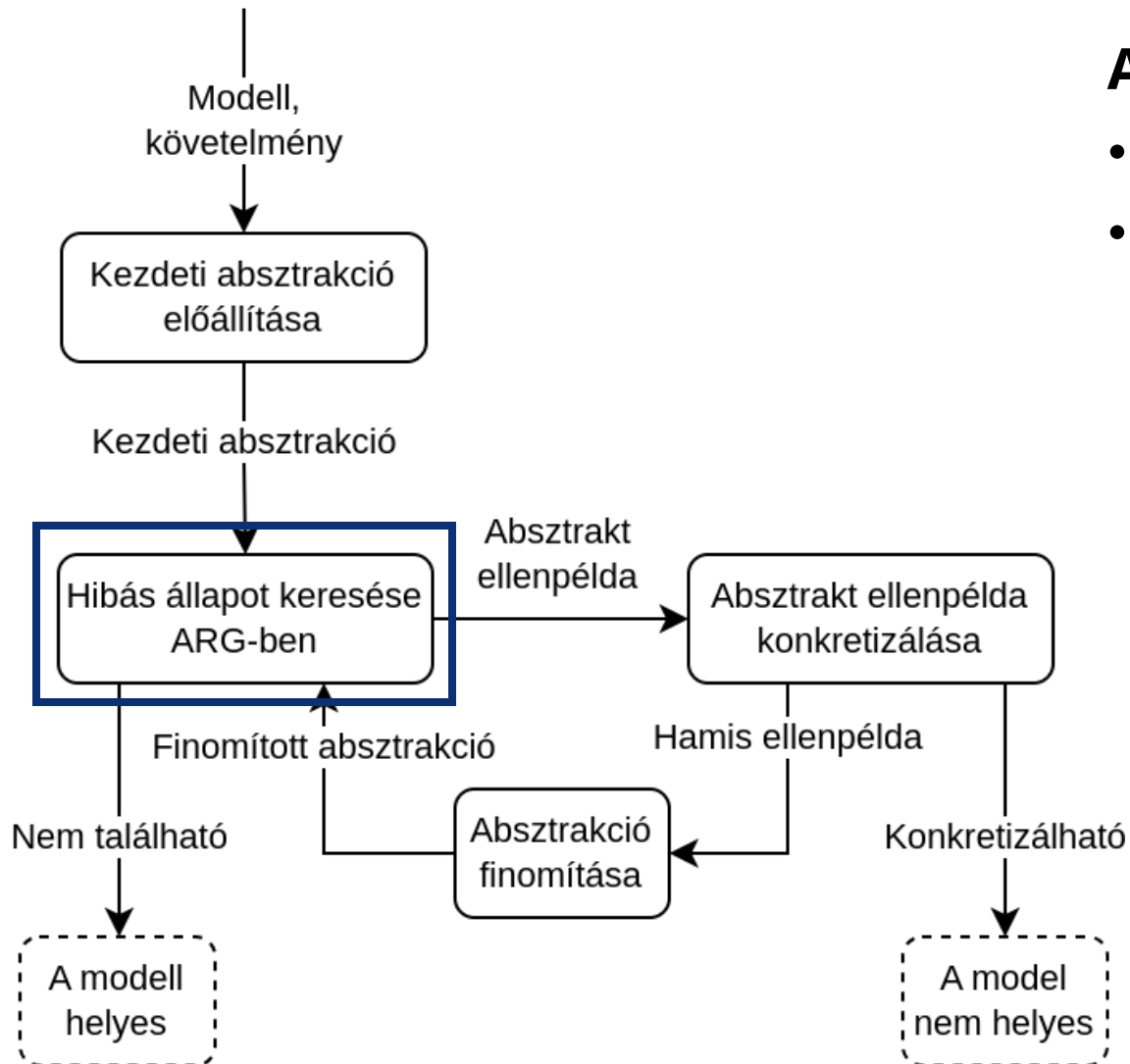
# CEGAR – Ellenpélda-vezérelt absztrakció finomítás



# CEGAR – Ellenpélda-vezérelt absztrakció finomítás

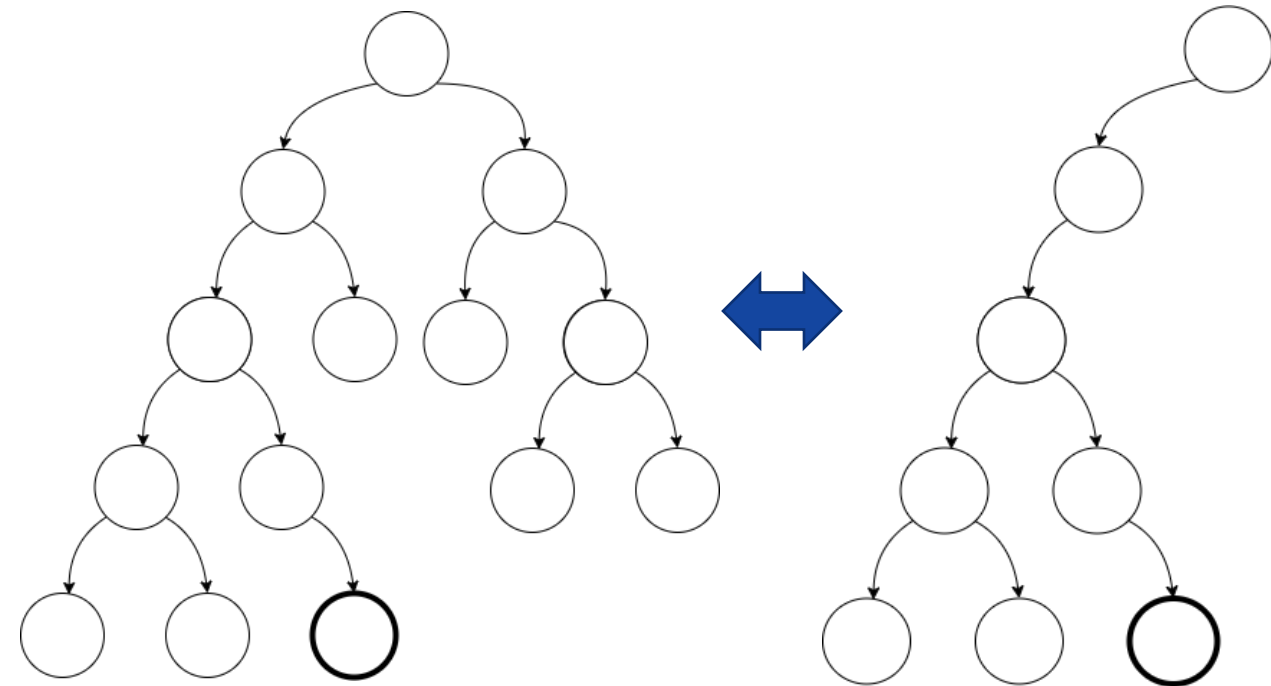


# CEGAR – Ellenpélda-vezérelt absztrakció finomítás

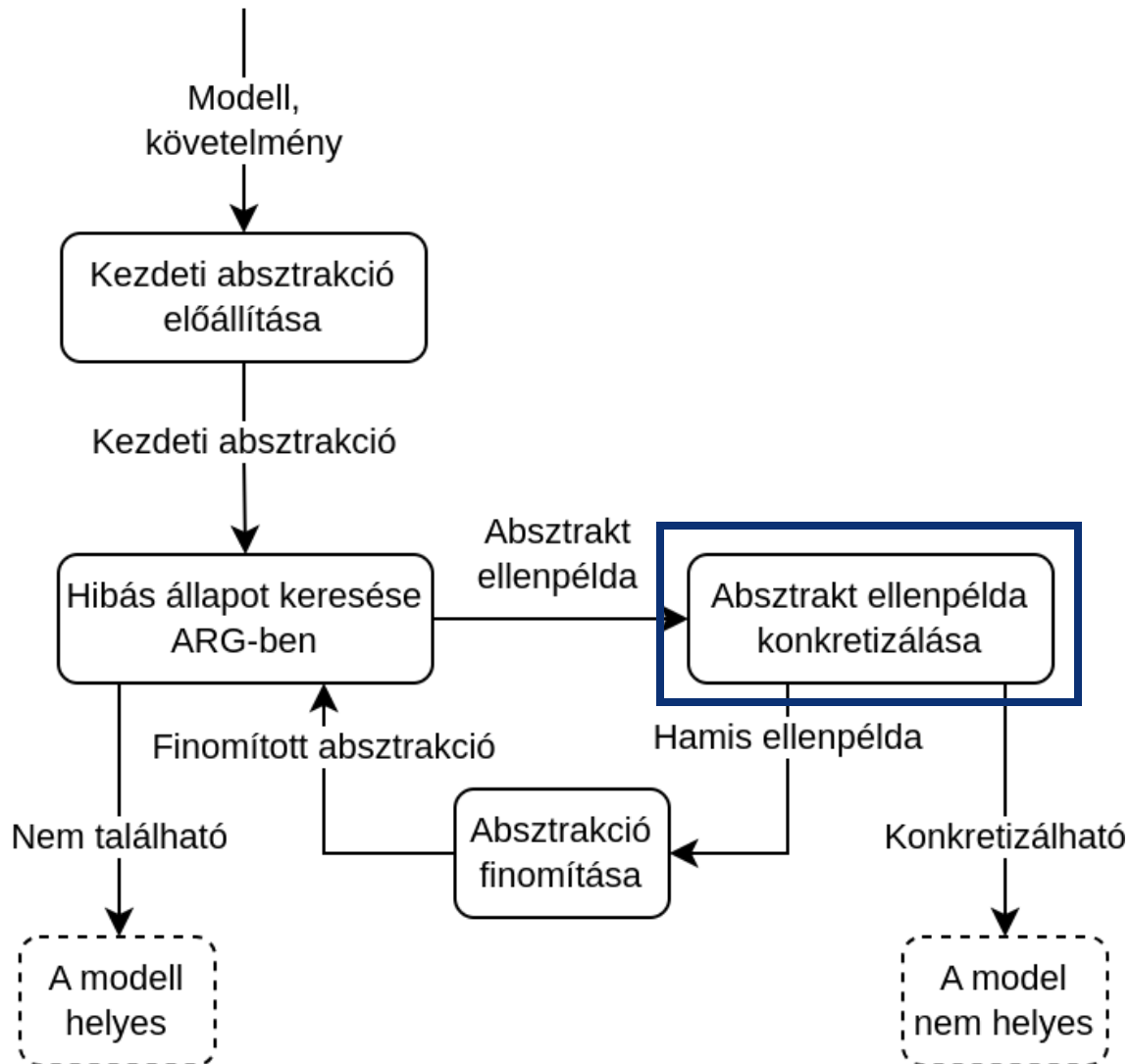


**ARG** - Absztrakt elérhetőségi gráf

- Adott absztrakt állapottér tömör reprezentációja
- Hibás állapotok **keresése**: BFS, DFS, heurisztikák

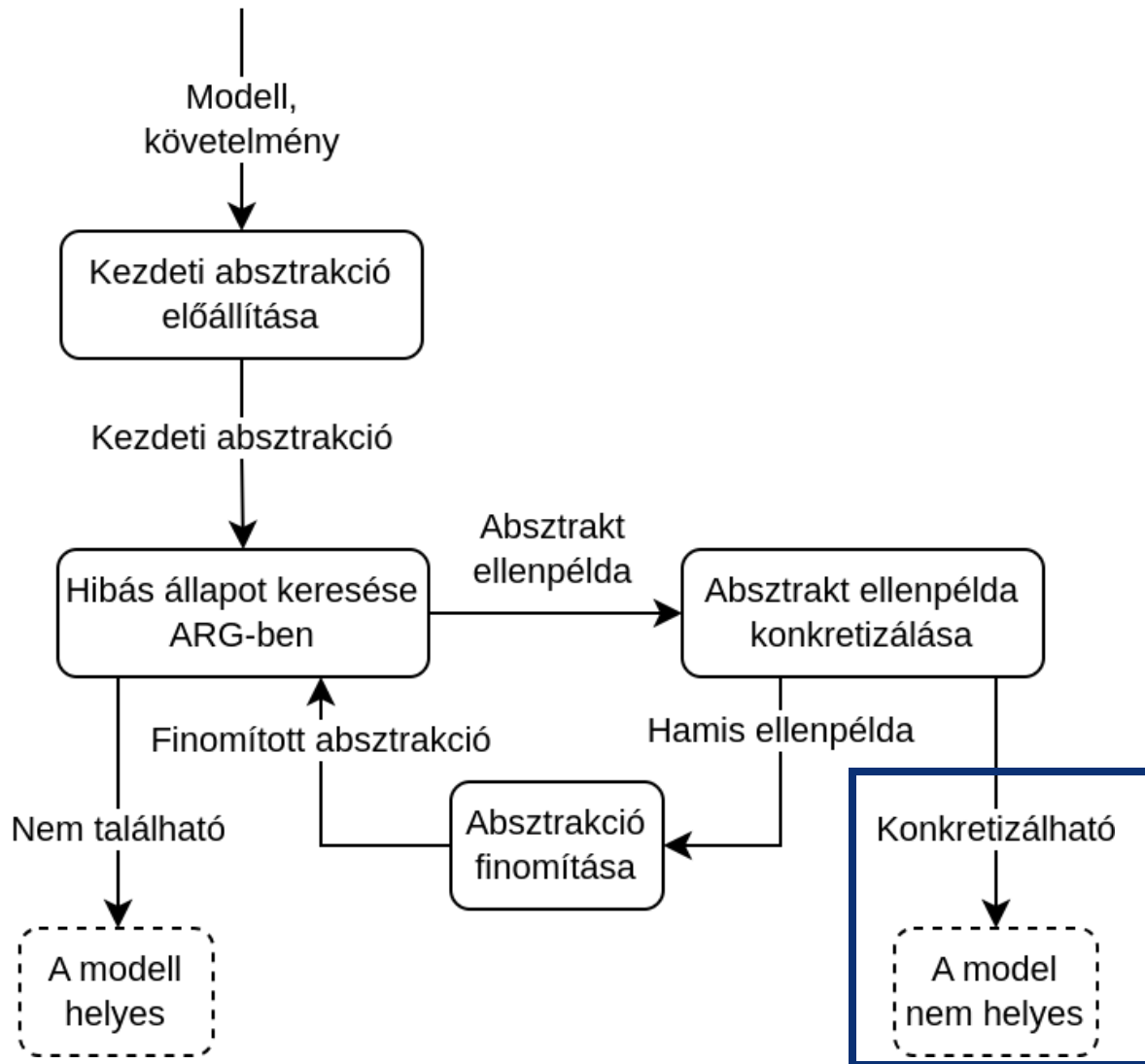


# CEGAR – Ellenpélda-vezérelt absztrakció finomítás

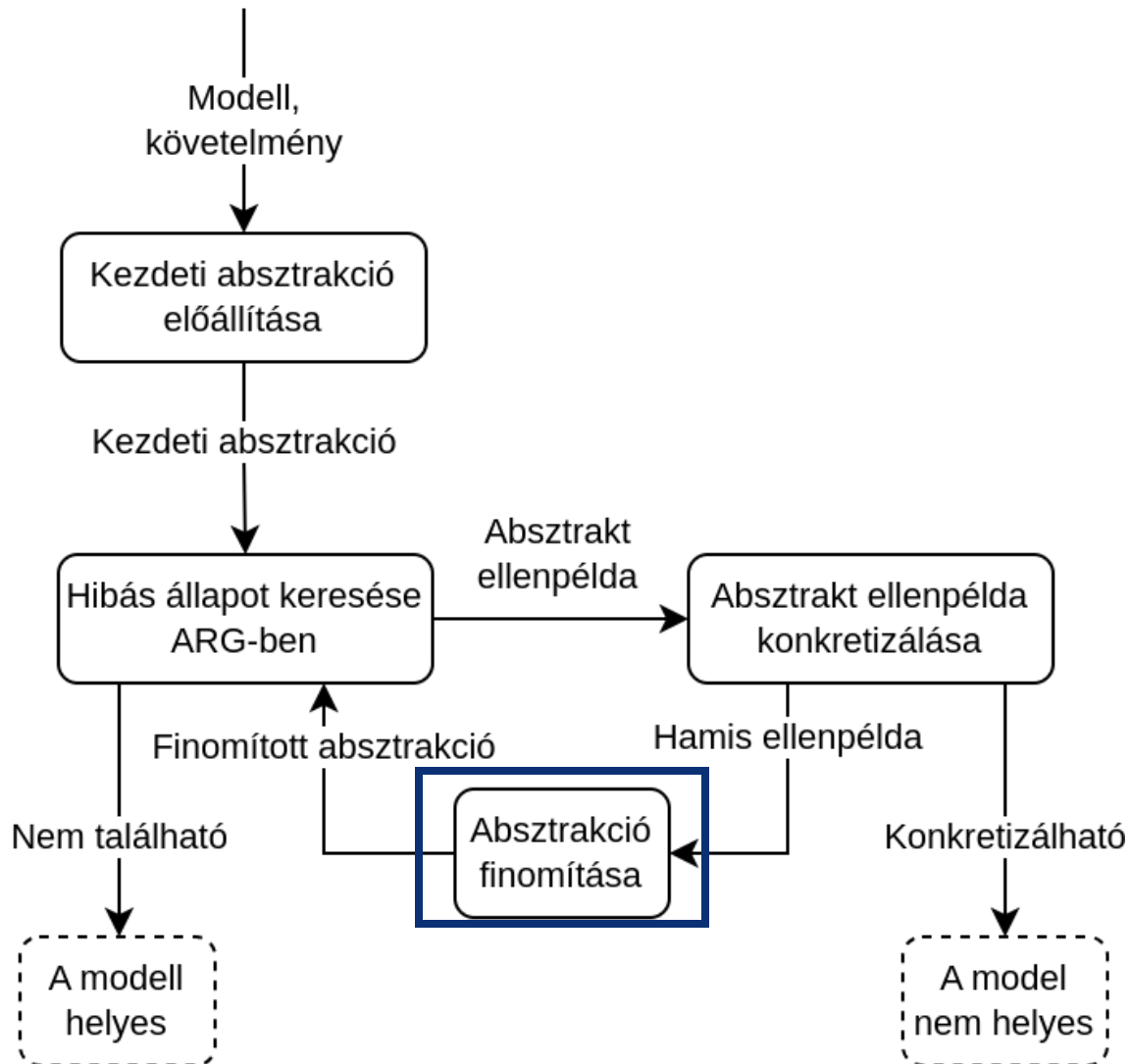




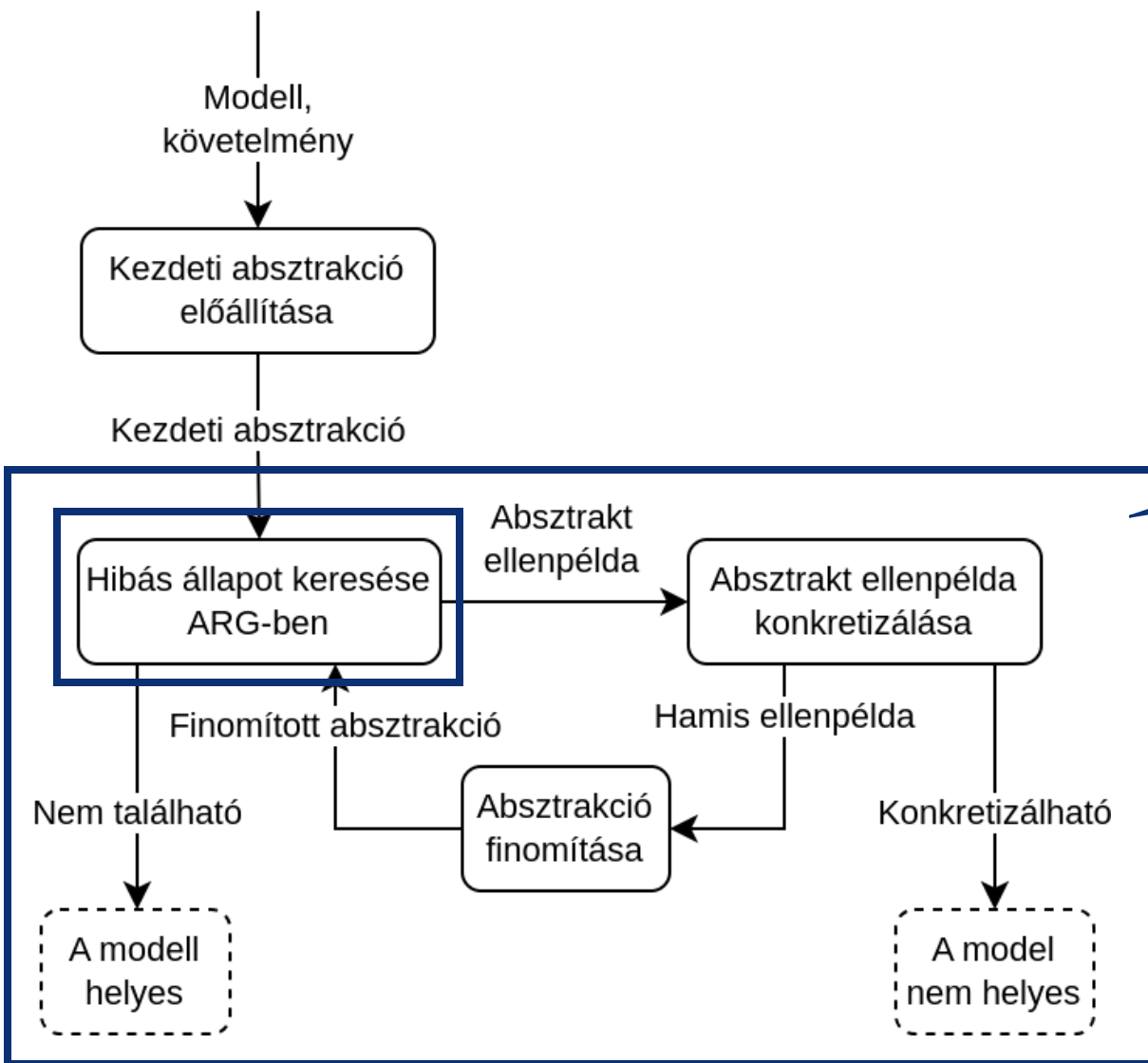
# CEGAR – Ellenpélda-vezérelt absztrakció finomítás



# CEGAR – Ellenpélda-vezérelt absztrakció finomítás



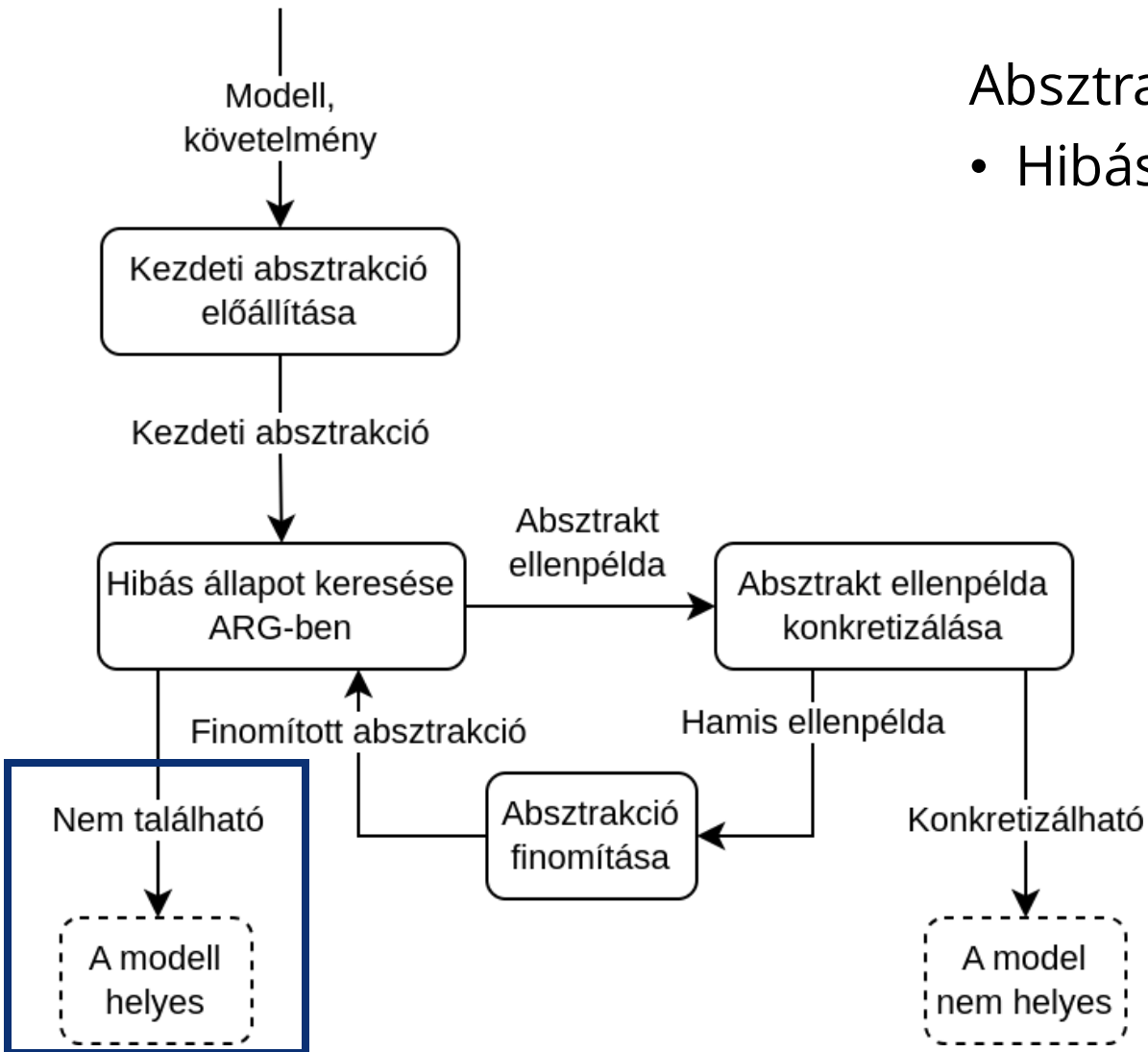
# CEGAR – Ellenpélda-vezérelt absztrakció finomítás



CEGAR hurok következő iterációja

# CEGAR – Ellenpélda-vezérelt absztrakció finomítás

- Absztrakt állapottér a lehetséges lefutásokat felülbecsli
- Hibás állapot nem érhető el → Helyes a rendszer

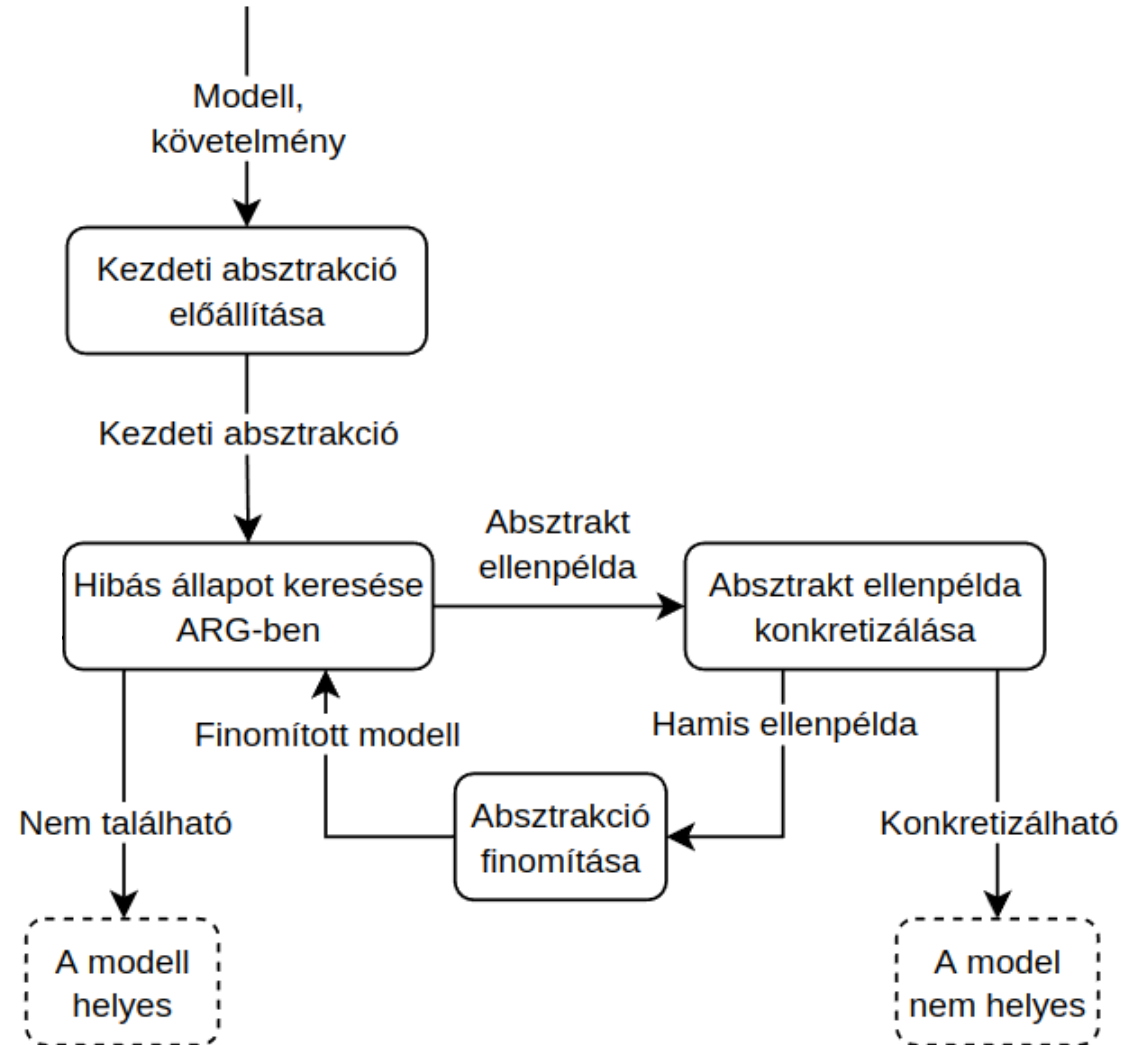


# Korábbi munkám

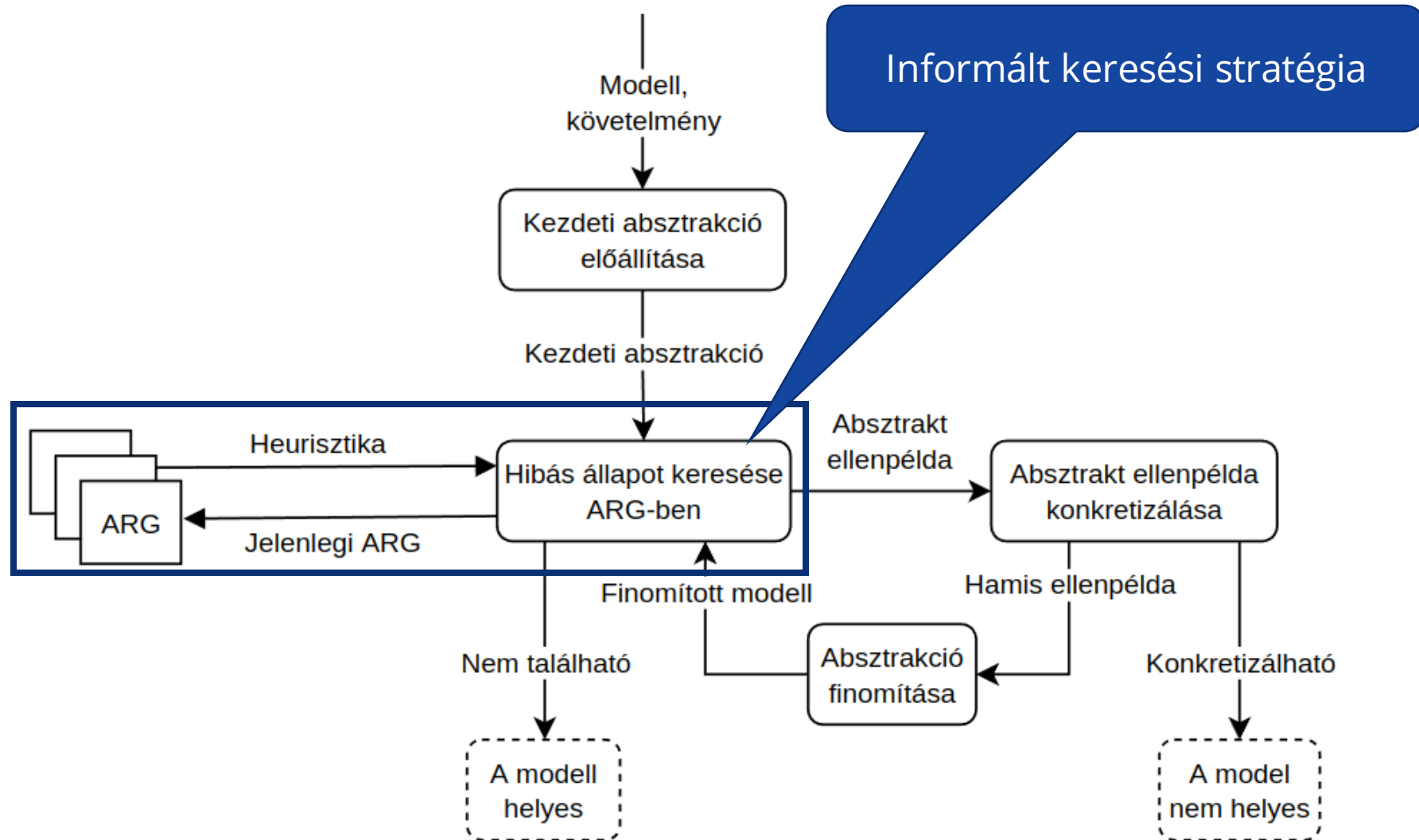
- Hatékony keresési algoritmusok kidolgozása
  - **Gyorsabb** helyesség bizonyítás
  - Hibás rendszer: **minimális hosszúságú ellenpélda**
- A\* keresés az absztrakt állapottérben
  - ARG-kből **távolság információ származtatása**
  - **Későbbi iterációbeli keresés támogatása heurisztika formájában**



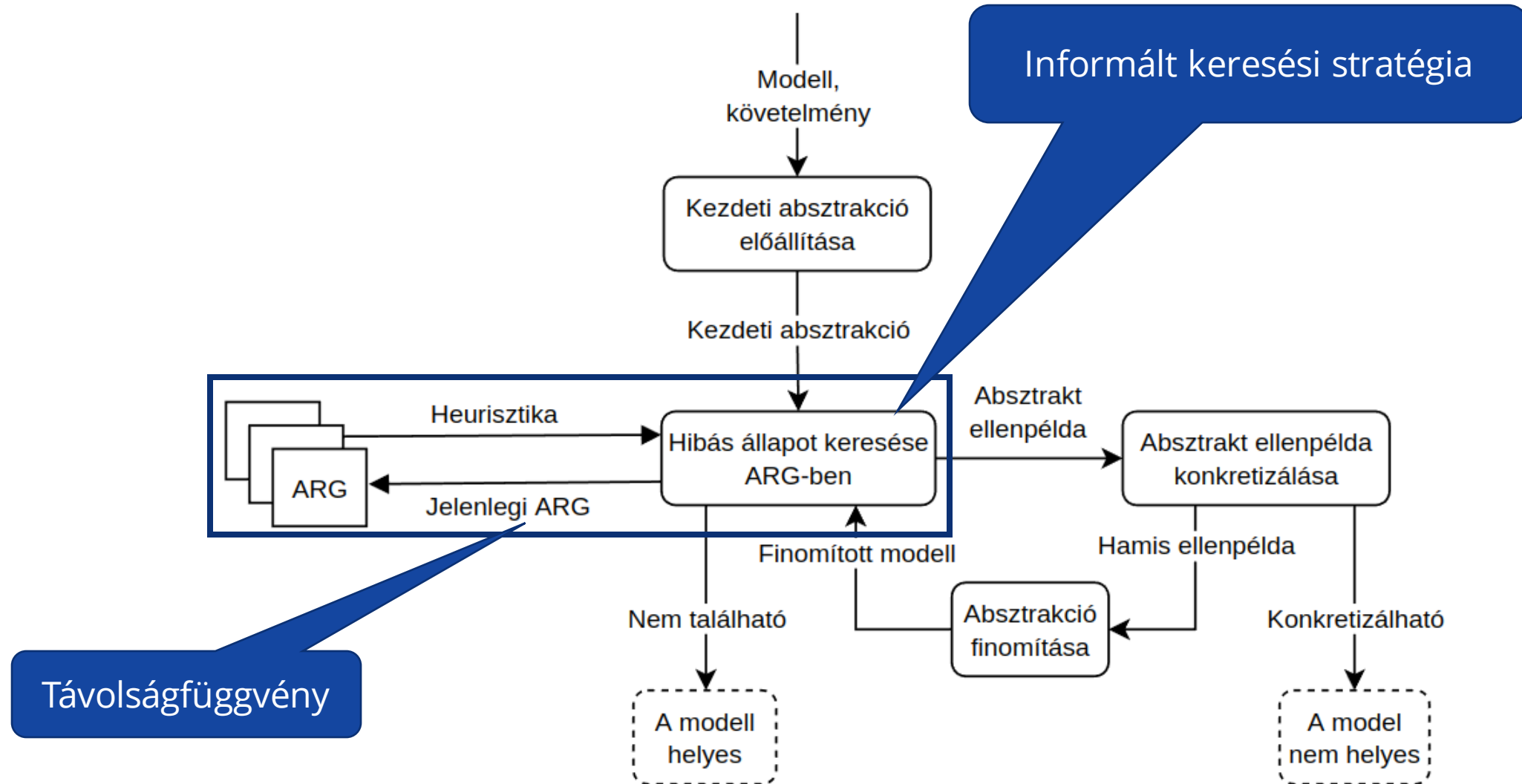
# A\* kereséssel támogatott CEGAR



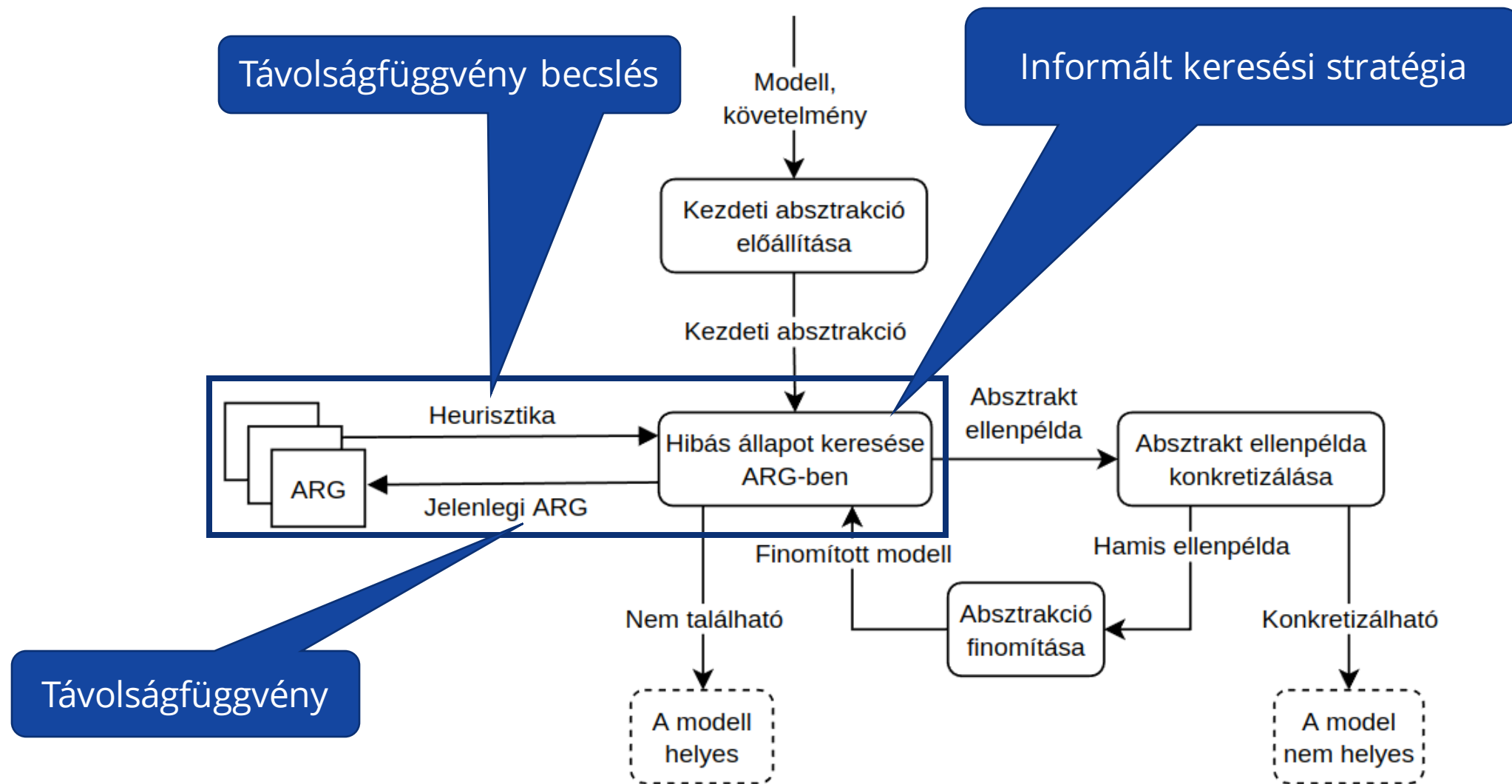
# A\* kereséssel támogatott CEGAR



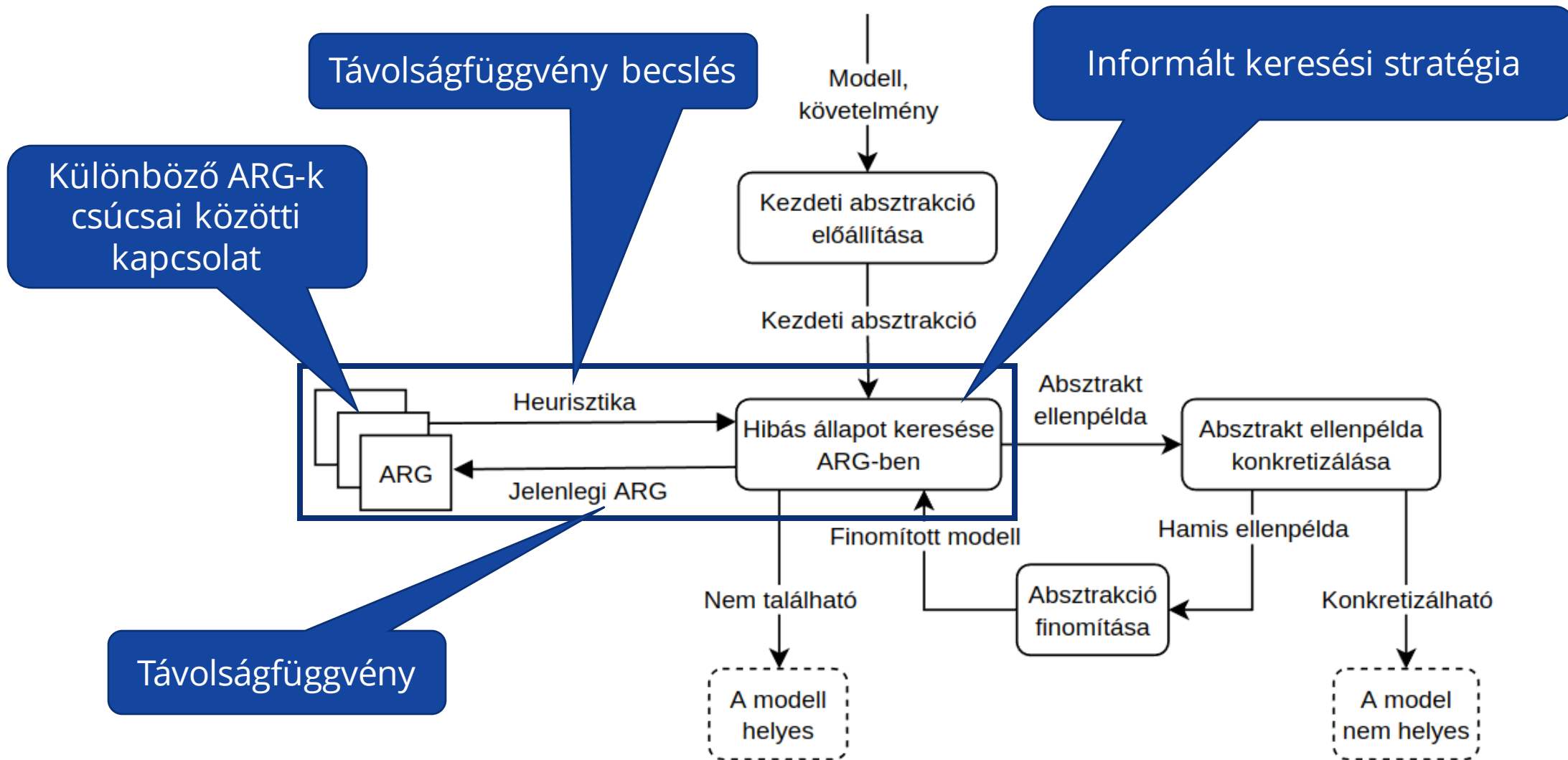
# A\* kereséssel támogatott CEGAR



# A\* kereséssel támogatott CEGAR

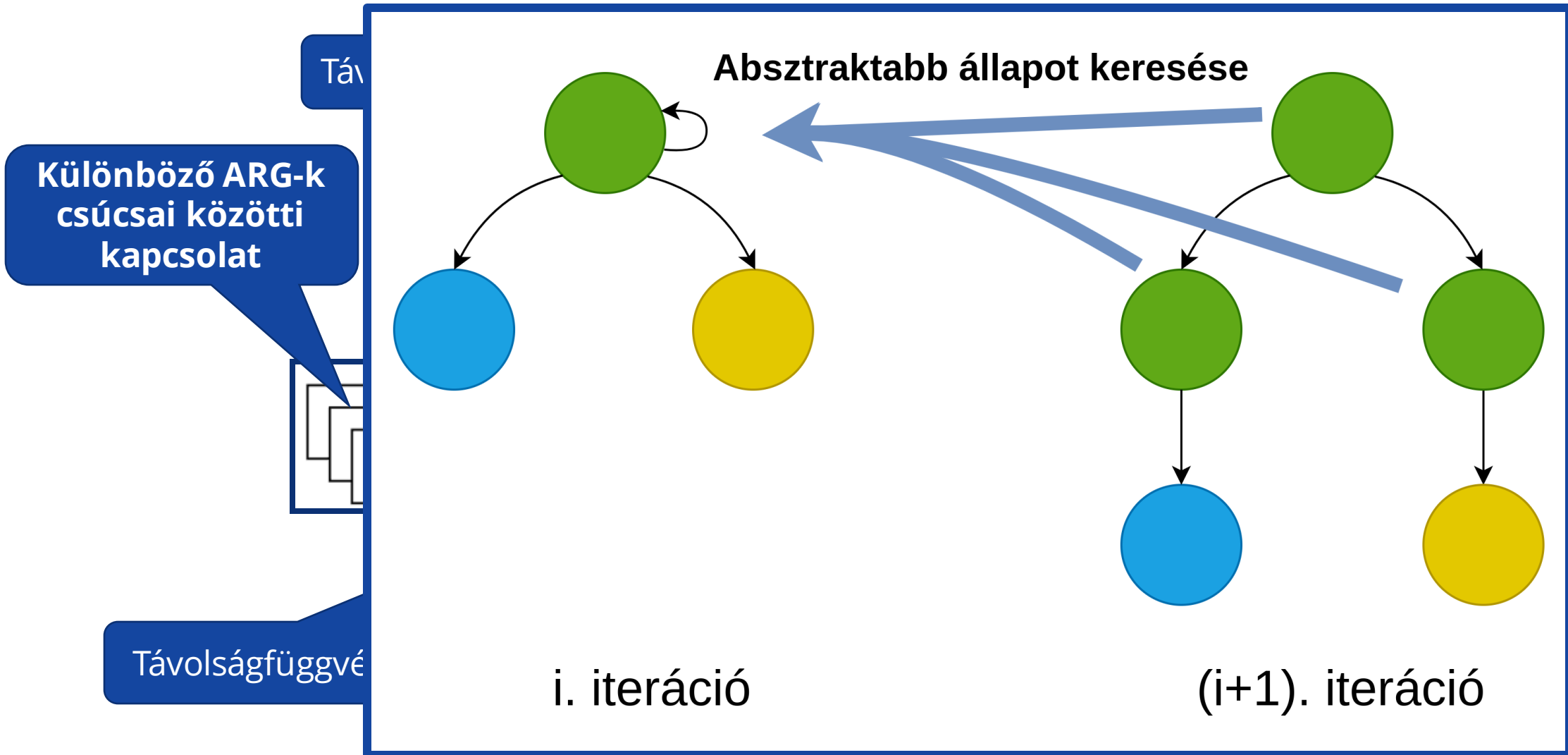


# A\* kereséssel támogatott CEGAR



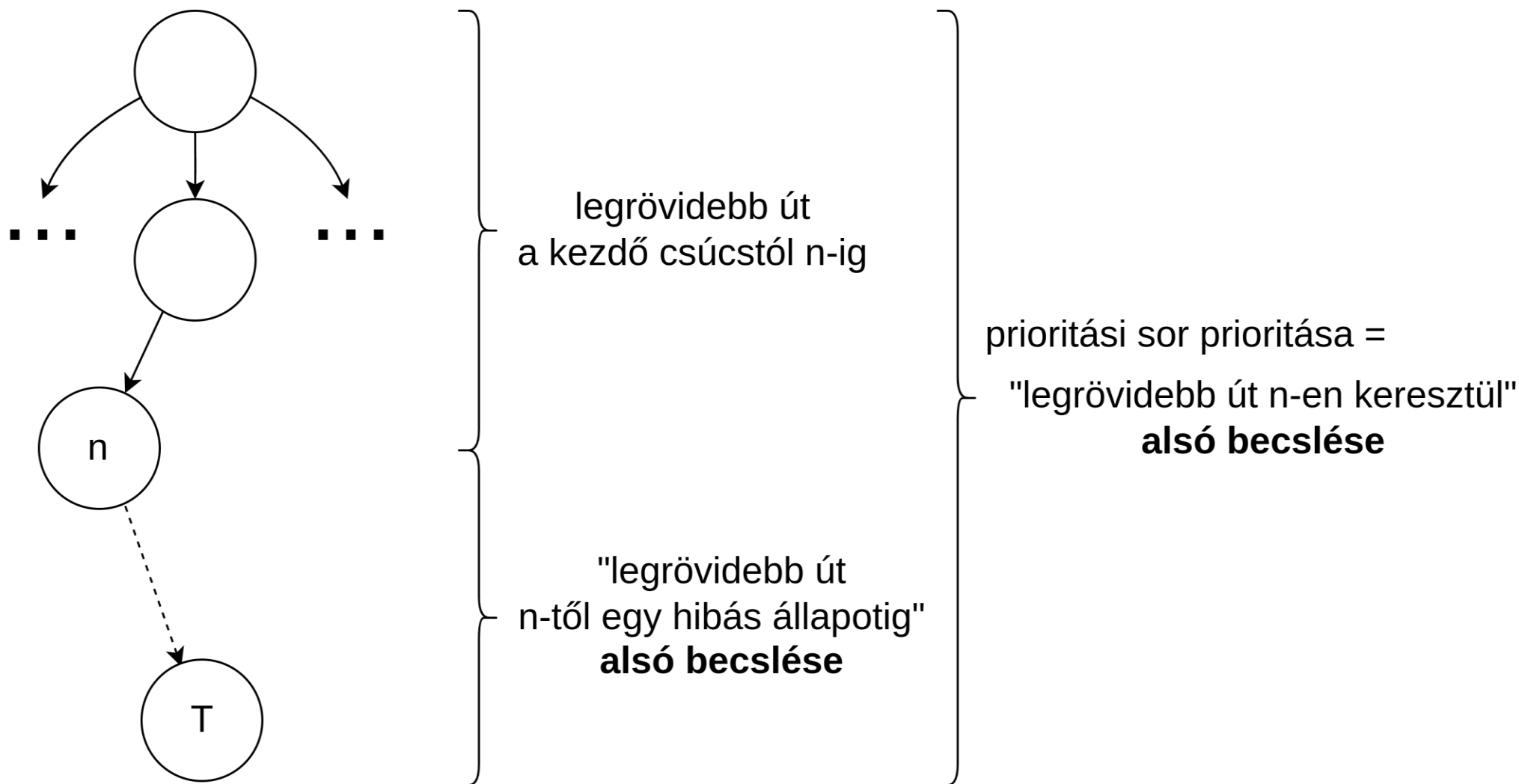


# A\* kereséssel támogatott CEGAR

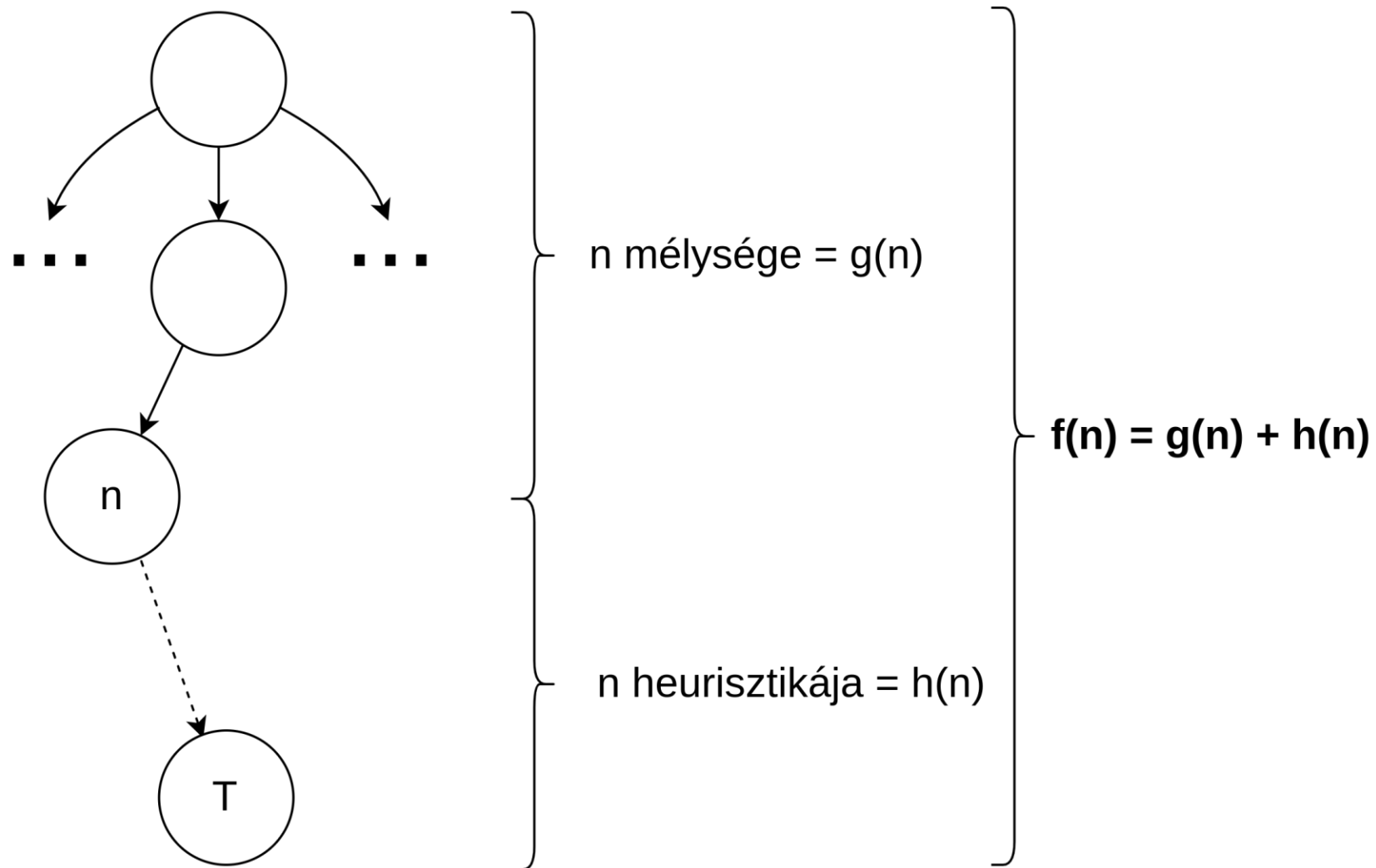


# Teljesen igény szerinti Hierarchikus A\*

# A\* keresés - Prioritási sor

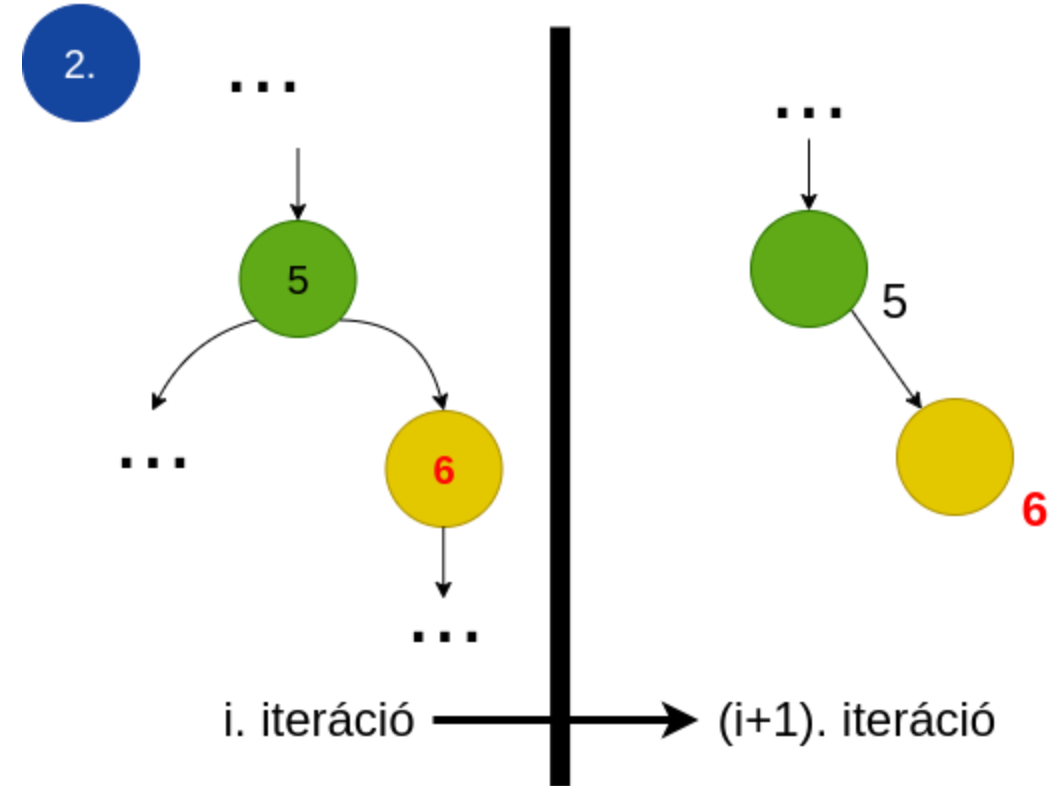
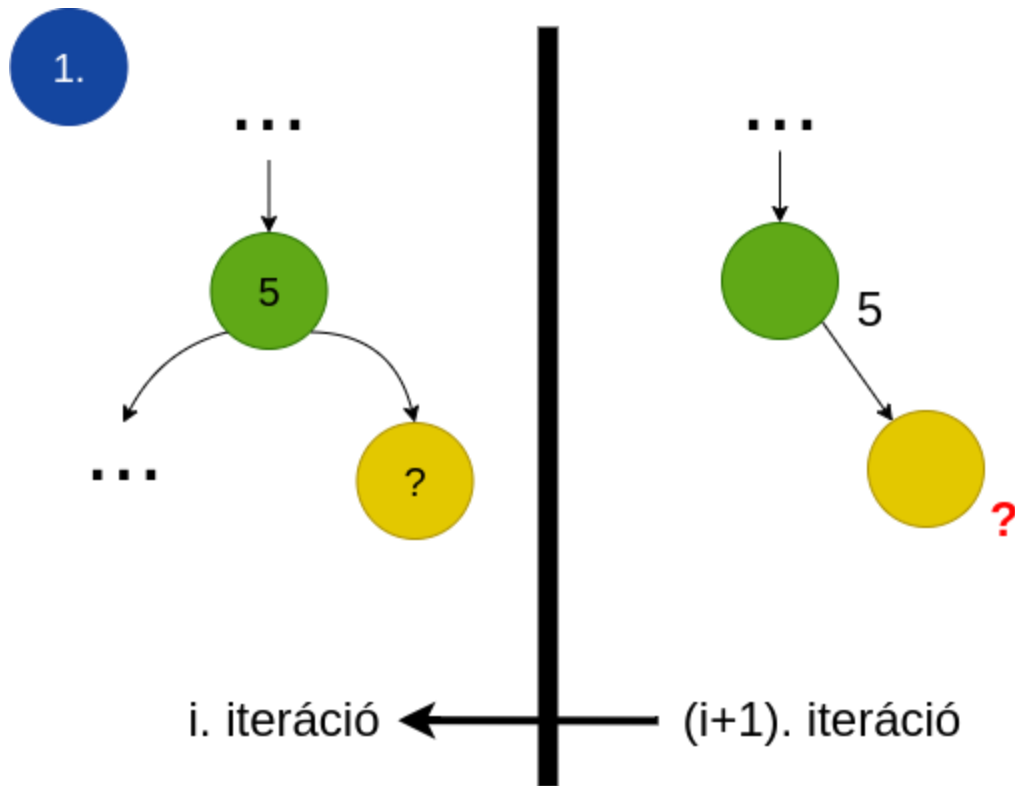


# A\* keresés - Prioritási sor



# Részlegesen igény szerinti Hierarchikus A\*

- Ha a korábbi ARG csúcsának távolsága (= **heurisztika**) **nem ismert**:
  - **új A\* keresés indítása** a csúctól annak távolságának megismerése érdekében





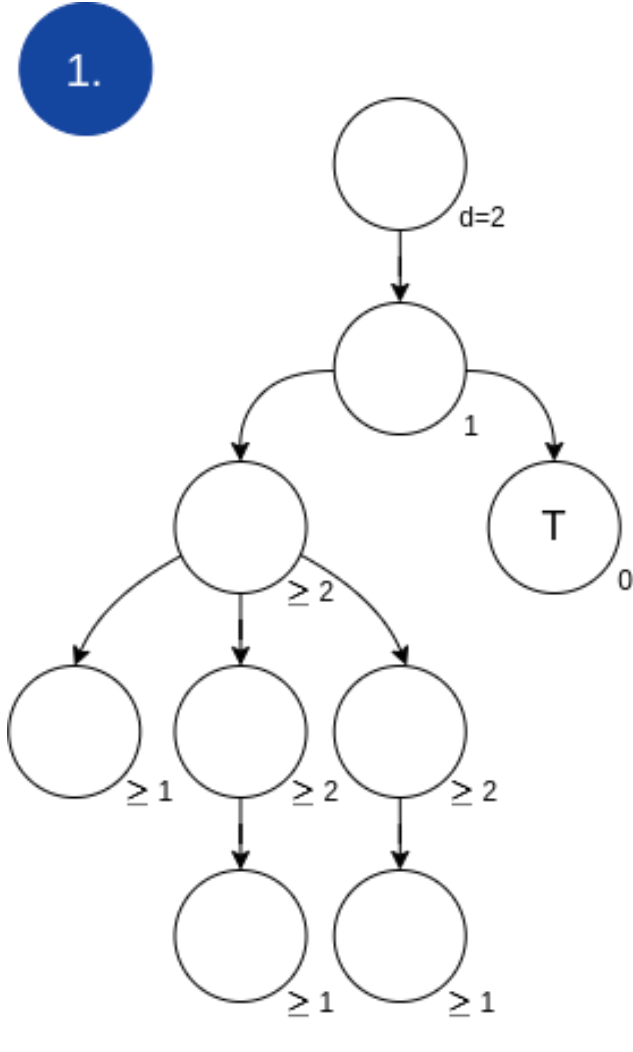
# Teljesen igény szerinti Hierarchikus A\*

$$f(n) = g(n) + h(n)$$

↑                      ↑  
mélység            heurisztika




- Ha a korábbi ARG csúcsának távolsága (= **heurisztika**) **nem ismert**:
  - ~~új A\* keresés indítása a csúcstól annak távolságának megismerése érdekében~~
  - Nem ismert heurisztika **prioritási sorba helyezése**
- A\* számára a sorból kivett elemnek a legkisebb f értékűnek kell lennie
  - Ezen követelmény teljesülése mellett **heurisztikák pontos megismerése elkerülhető**

# Teljesen igény szerinti Hierarchikus A\*



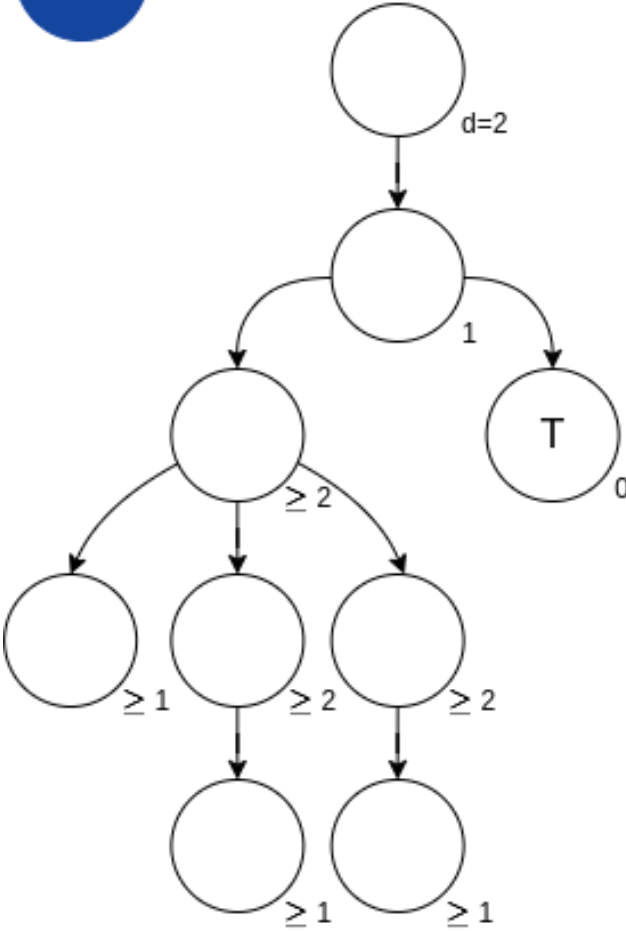
i+1. iteráció

$f(n) = g(n) + h(n)$   
          ↑          ↑  
      mélység  heurisztika

-  Aktív elem
-  Sorban lévő elem
-  Hibás állapot

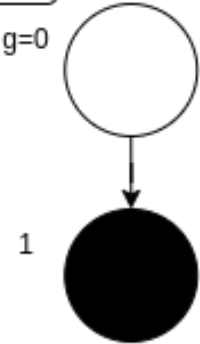
# Teljesen igény szerinti Hierarchikus A\*

2.




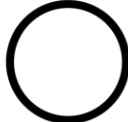

i. iteráció

Sor: 2



i+1. iteráció

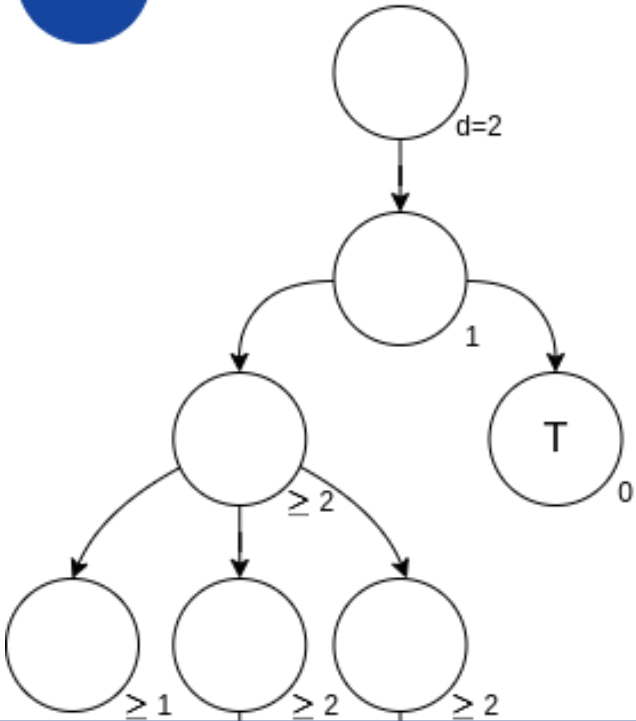
$f(n) = g(n) + h(n)$   
          ↑          ↑  
      mélység  heurisztika

-  Aktív elem
-  Sorban lévő elem
-  Hibás állapot

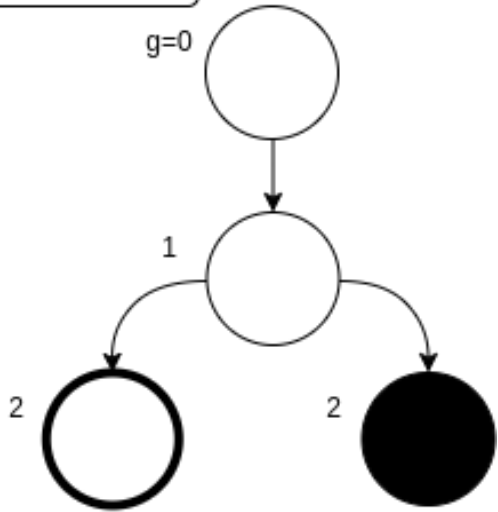
# Teljesen igény szerinti Hierarchikus A\*

$f(n) = g(n) + h(n)$   
↑            ↑  
mélység    heurisztika

3.



Sor: 2, ≥ 4



- Aktív elem
- Sorban lévő elem
- T Hibás állapot

Ha az egyik legkisebb elem pontos heurisztikával rendelkezik  
- Az a legkisebb f értékű elem

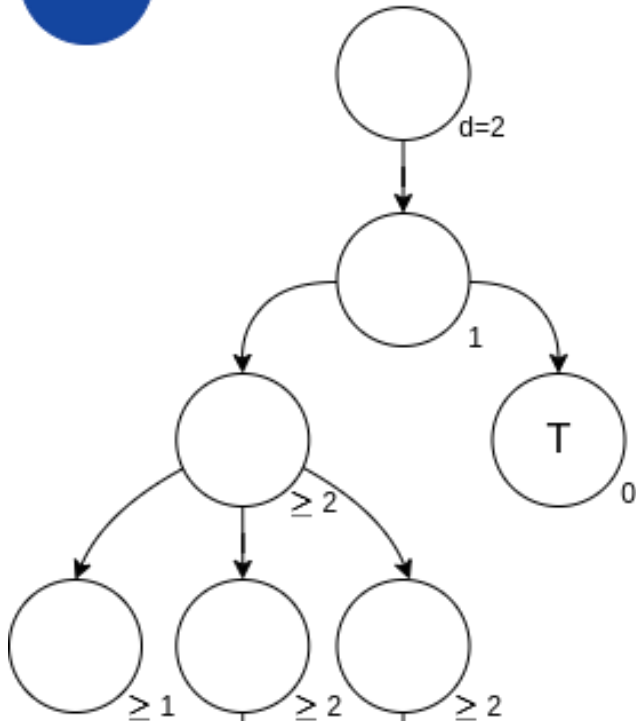
i. iteráció

i+1. iteráció

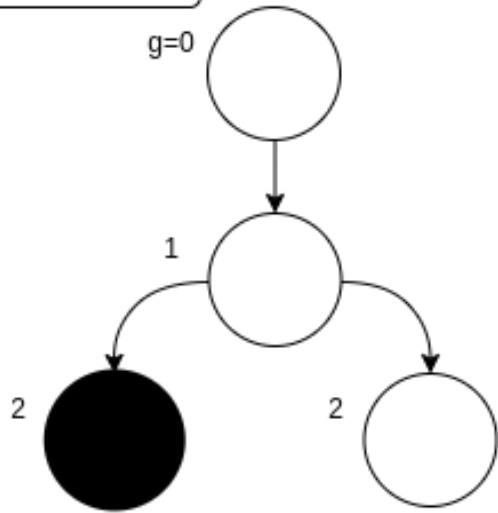
# Teljesen igény szerinti Hierarchikus A\*


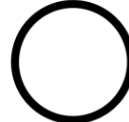
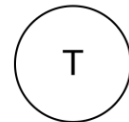
$f(n) = g(n) + h(n)$   
↑            ↑  
mélység    heurisztika

4.



Sor:  $\geq 4$



-  Aktív elem
-  Sorban lévő elem
-  Hibás állapot

Egy elemű a sor  
- Még ha nem is pontos a heurisztika, ez a legkisebb elem

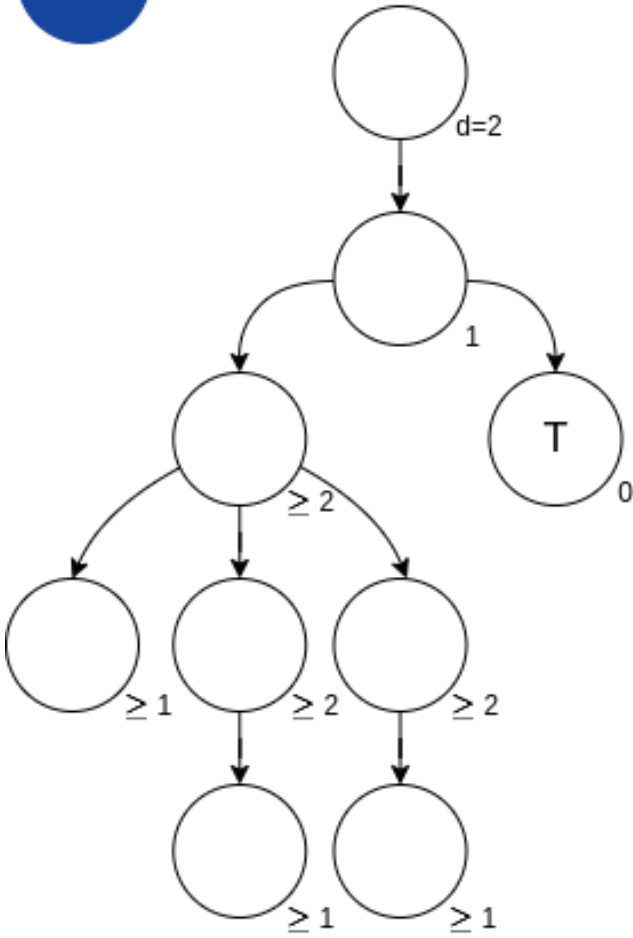
i. iteráció

i+1. iteráció

# Teljesen igény szerinti Hierarchikus A\*

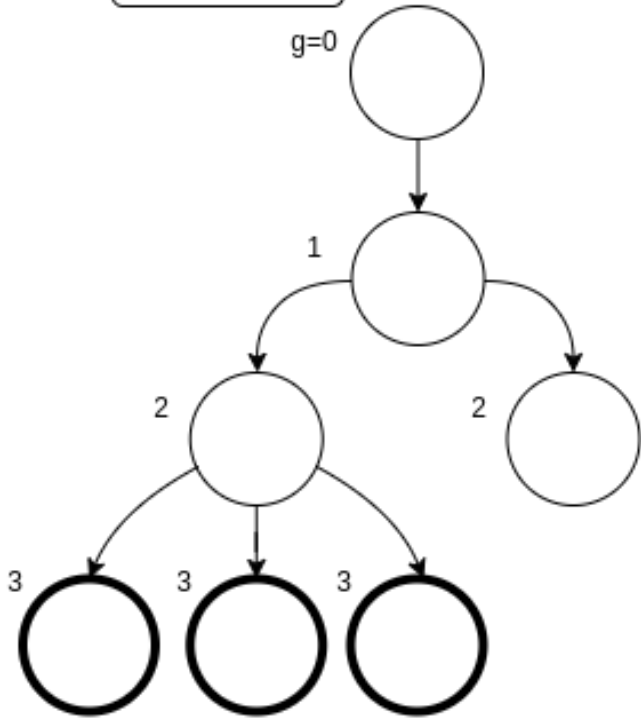
$f(n) = g(n) + h(n)$   
↑            ↑  
mélység    heurisztika

5.



$i$ . iteráció

Sor:  $\geq 4, \geq 5, \geq 5$



$i+1$ . iteráció

- Aktív elem
- Sorban lévő elem
- Hibás állapot

# Teljesen igény szerinti Hierarchikus A\*

$$f(n) = g(n) + h(n)$$

↑            ↑  
mélység    heurisztika

6.

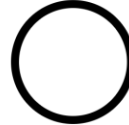
Sor:  $\geq 4, \geq 5, \geq 5$

$g=0$

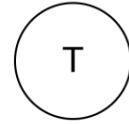
A két legkisebb elem nem pontos heurisztikával rendelkezik  
- Legkisebb elem pontosítása a "második elem f értéke" + 1-ig



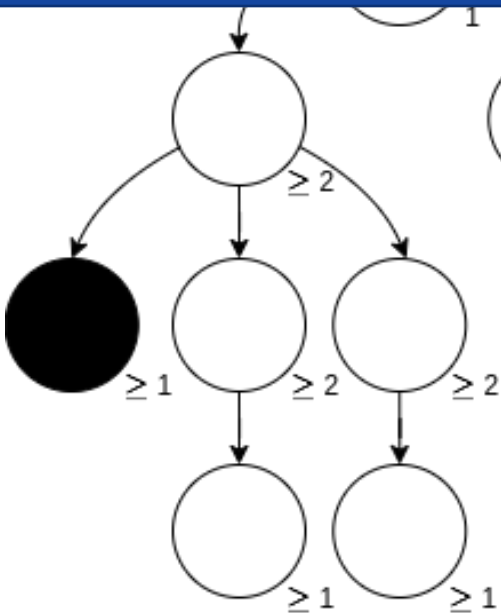
Aktív elem



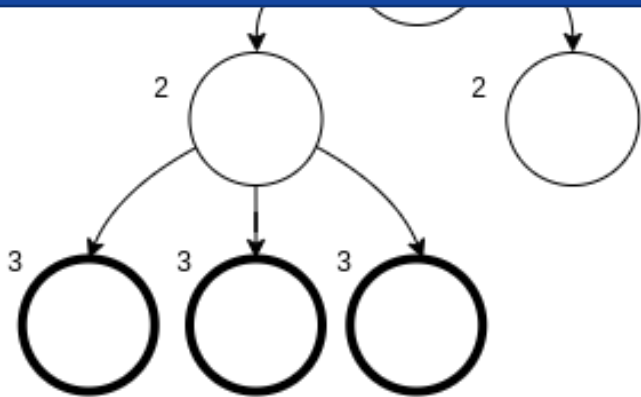
Sorban lévő elem



Hibás állapot



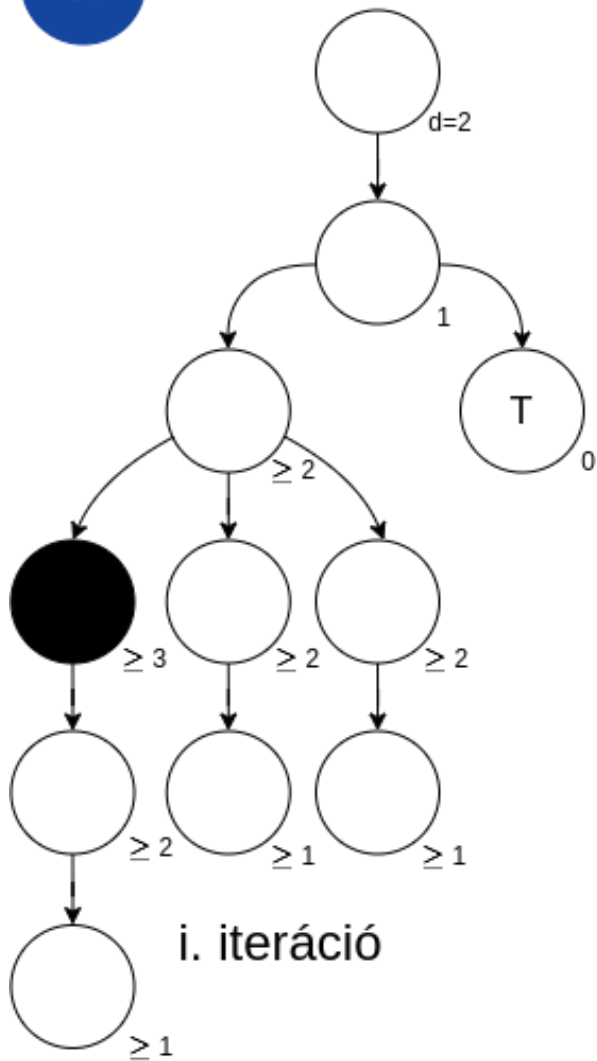
i. iteráció



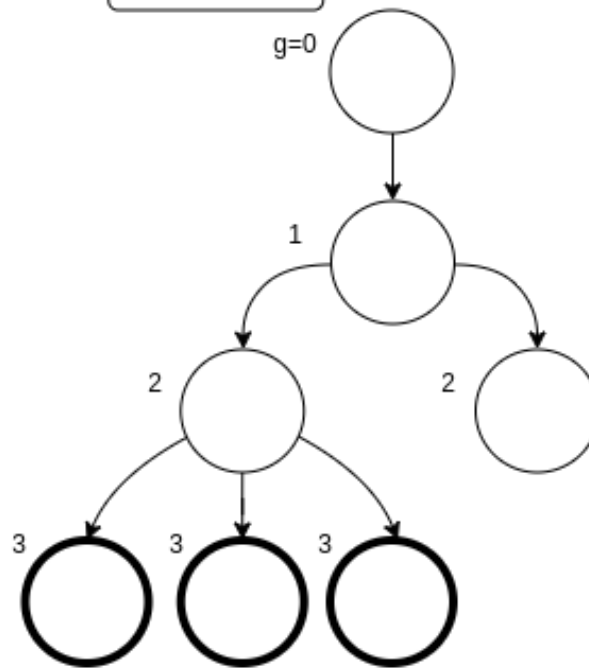
i+1. iteráció

# Teljesen igény szerinti Hierarchikus A\*

7.



**Sor:**  $\geq 5, \geq 5, \geq 6$

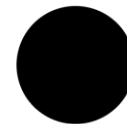


i+1. iteráció

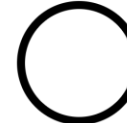
$$f(n) = g(n) + h(n)$$

mélység

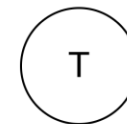
heurisztika



Aktív elem



Sorban lévő elem

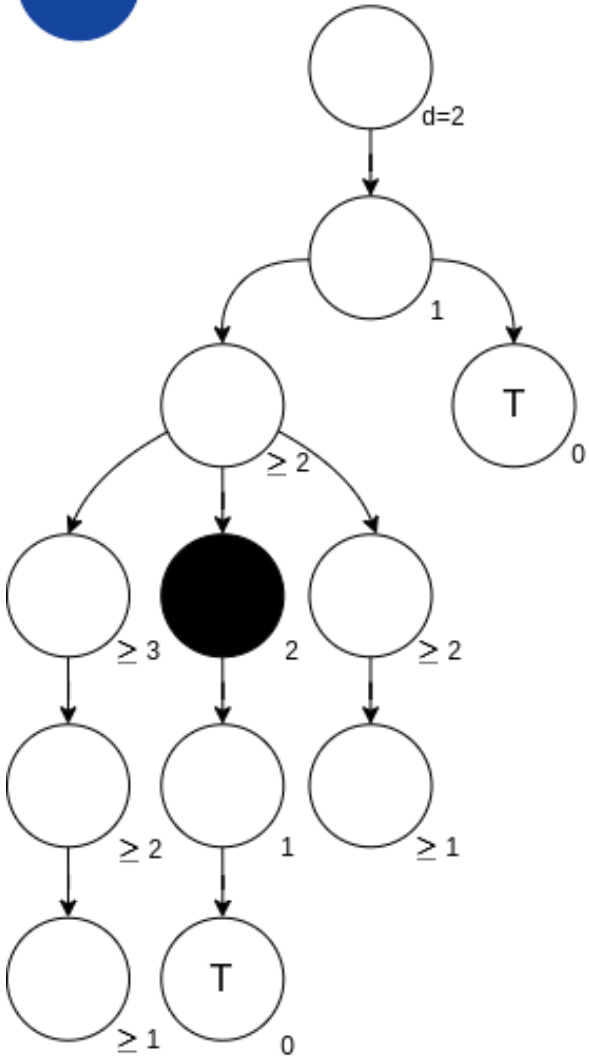


Hibás állapot

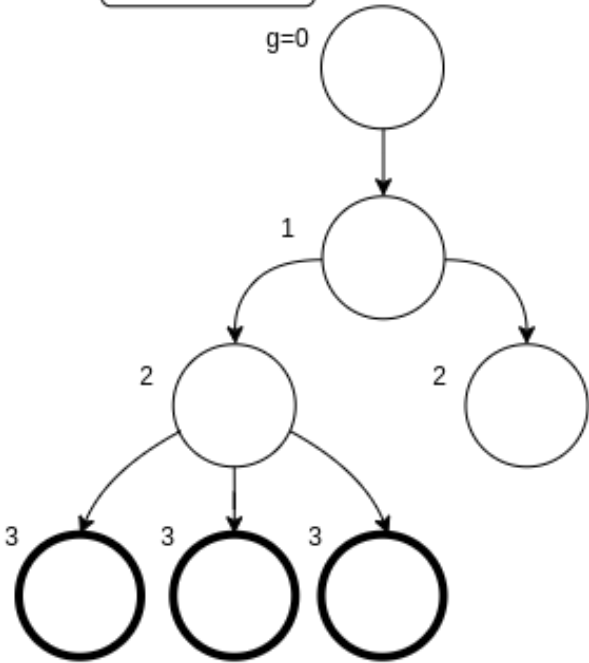


# Teljesen igény szerinti Hierarchikus A\*

8.


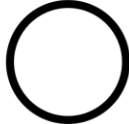



Sor: 5,  $\geq 5$ ,  $\geq 6$



$i+1$ . iteráció

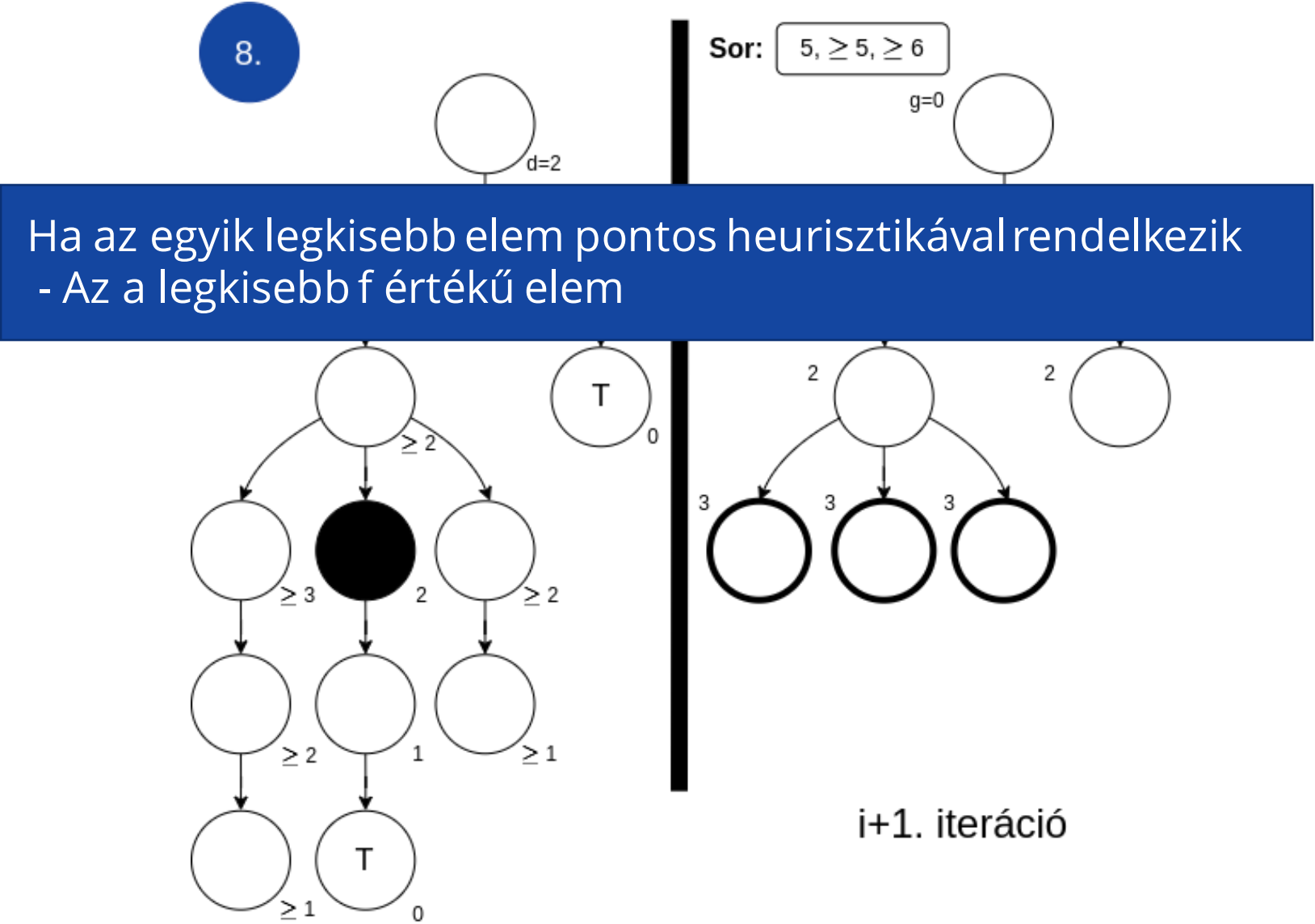
$f(n) = g(n) + h(n)$   
          ↑          ↑  
      mélység  heurisztika

-  Aktív elem
-  Sorban lévő elem
-  Hibás állapot

# Teljesen igény szerinti Hierarchikus A\*

$$f(n) = g(n) + h(n)$$

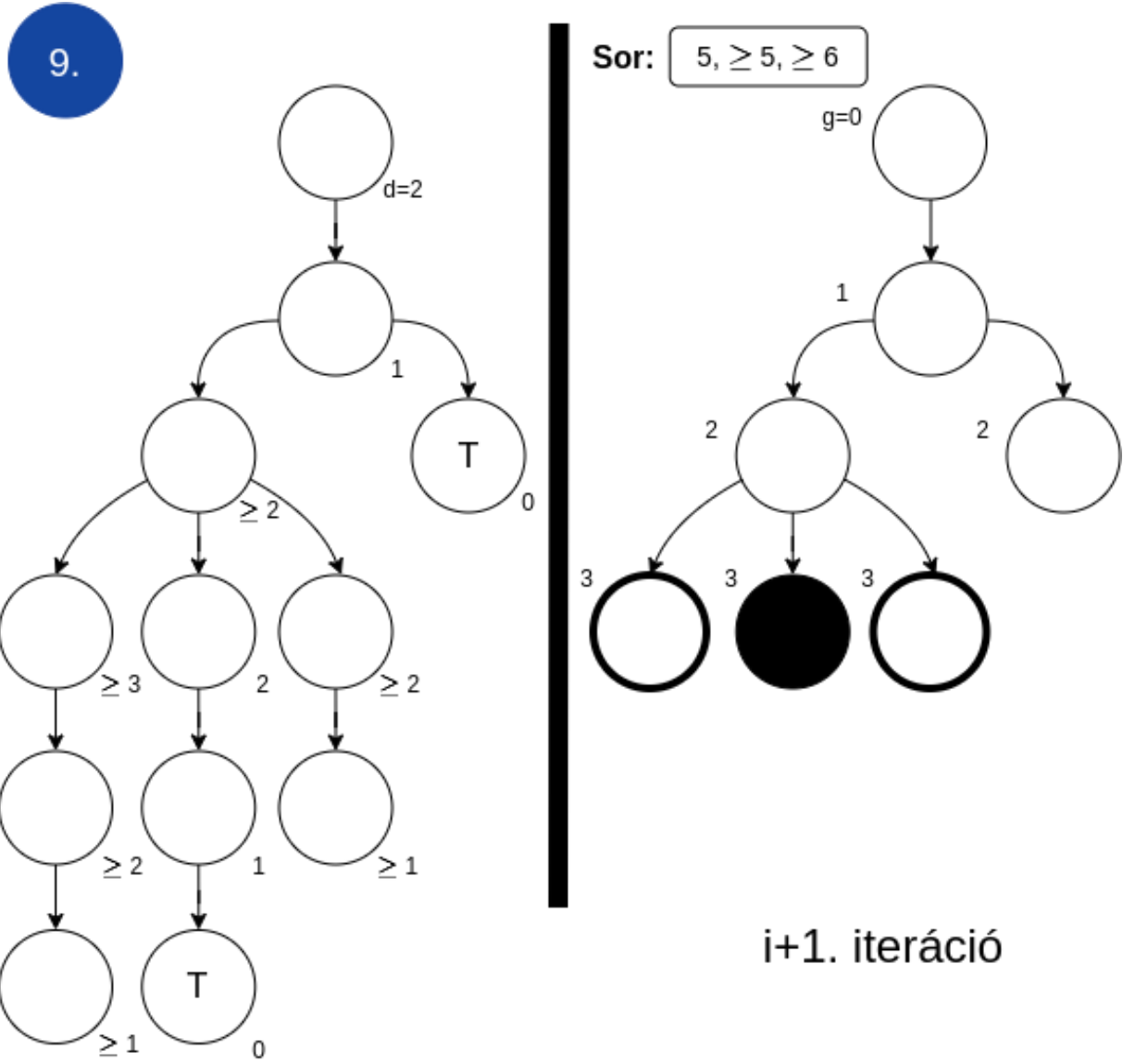
↑            ↑  
mélység    heurisztika



- Aktív elem
- Sorban lévő elem
- Hibás állapot

# Teljesen igény szerinti Hierarchikus A\*

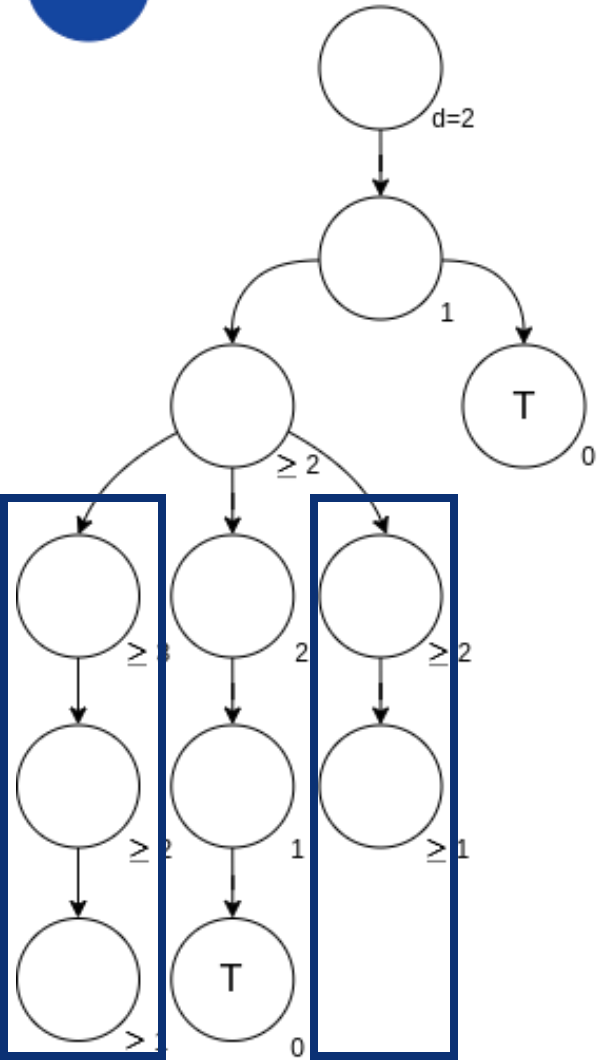
$f(n) = g(n) + h(n)$   
↑            ↑  
mélység    heurisztika



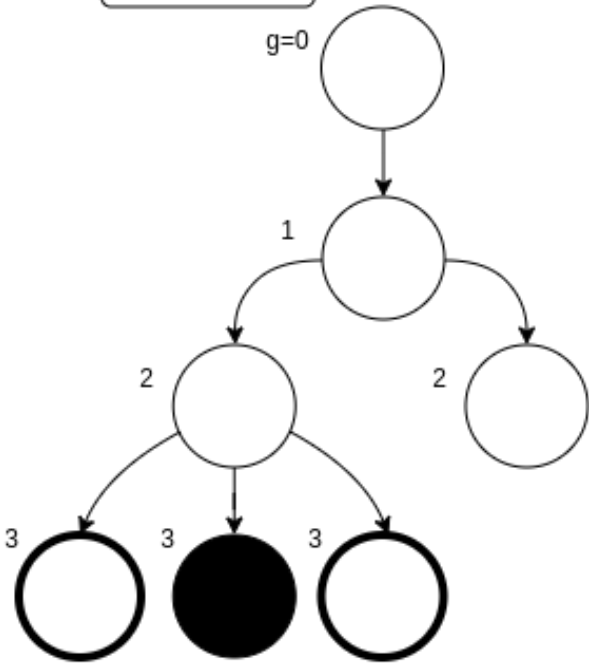
# Teljesen igény szerinti Hierarchikus A\*

$f(n) = g(n) + h(n)$   
↑            ↑  
mélység    heurisztika


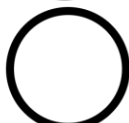
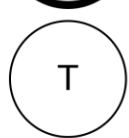
9.



Sor: 5, ≥ 5, ≥ 6



i+1. iteráció

-  Aktív elem
-  Sorban lévő elem
-  Hibás állapot

# Implementáció

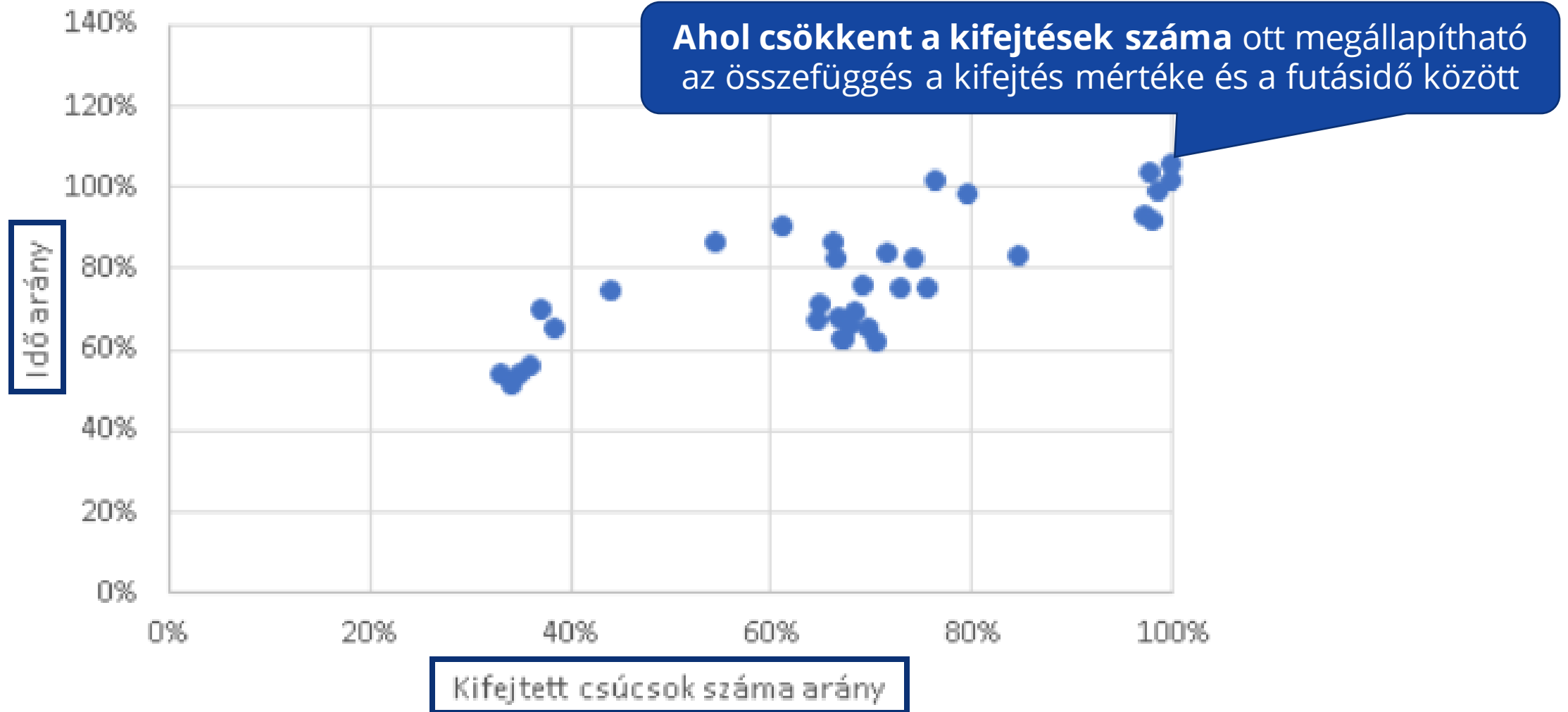
- Korábbi architektúra stabilizálása a **nyílt forráskódú** Thetában
- Könnyen kiegészíthető újabb Hierarchikus A\* változatokkal
- Teljesen igény szerinti Hierarchikus A\* implementálása



## Mérések

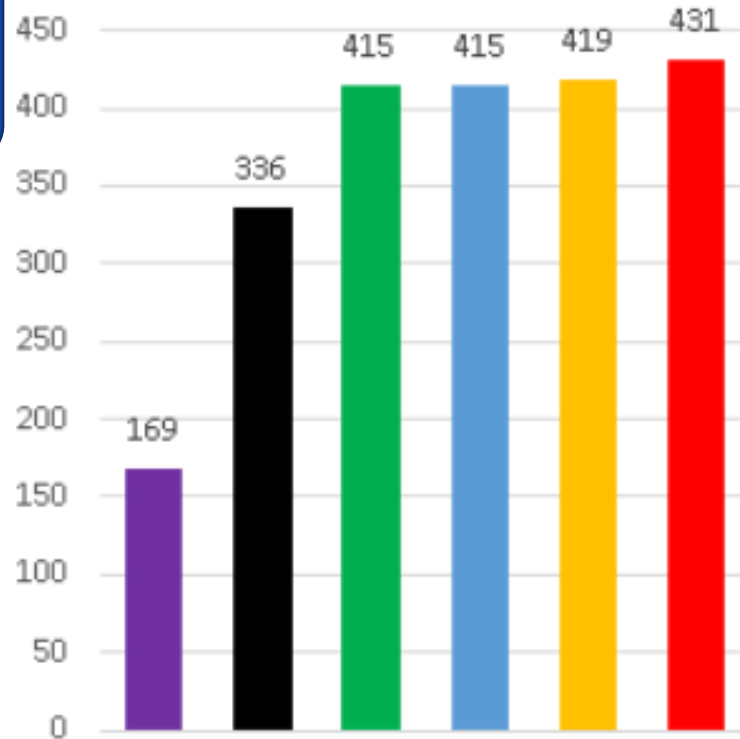
- Ipari modellek verifikálása, különböző konfigurációk mellett:
  - Keresési algoritmusok
    - Korábbi Hierarchikus A\* változatok + Teljesen igény szerinti változat
    - BFS
    - ERR: egyszerű heurisztika modell szerkezete alapján
  - Absztrakciók
  - ...

# Mérések: Részlegesen igény szerinti és Teljesen igény szerinti összehasonlítása



# Mérések: Időkorláton belül elvégzett verifikációk száma

Általánosságban elmondható, hogy a többi keresési stratégiához képest rosszabbul teljesít

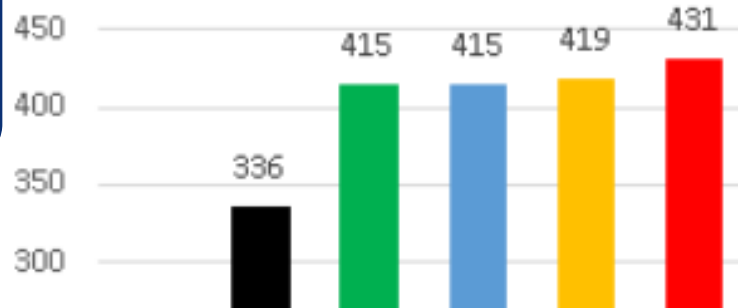


● A\* Teljesen igény szerinti  
● A\* Részlegesen igény szerinti

● ERR ● BFS  
● A\* Csökkentékes  
● A\* Teljes

# Mérések: Időkorláton belül elvégzett verifikációk száma

Általánosságban elmondható, hogy a többi keresési stratégiához képest rosszabbul teljesít



Mivel a legtöbb modellnél **nem volt lehetséges kevesebb csúcsot kifejteni**, így a **keresések megszakításának overheadje** nem lett ellensúlyozva.



● A\* Teljesen igény szerinti  
● A\* Részlegesen igény szerinti

● ERR ● BFS  
● A\* Csökkentéses  
● A\* Teljes



# Összefoglalás

- Elméleti eredményeim:
  - Teljesen igény szerinti Hierarchikus A\* változat kidolgozása
- Gyakorlati eredményeim:
  - Architektúra stabilizáció
  - Teljesen igény szerinti változat **nyílt forráskódú** implementációja a **Theta** modellellenőrző keretrendszerben
  - Teljesítmény kiértékelése verifikációk futtatásával



# Bírálóí kérdés 1.

Van-e olyan egyszerű példa, ahol a keresési algoritmus implementációjának működése igazolható?

Benchmarkolás előtt a Thetában található egyszerűbb, illetve nagyobb absztrakt állapotterű modelleken is ellenőrizve lett

Az algoritmus implementációjával szemben támasztott elméleti működést ellenőrző assertionök sikeresen lefutottak

# Bírálóí kérdés 2.

Vizsgálható-e a bemutatott eljárás saját magával?

Számos gyakorlati akadályba ütközik (Kotlin kód feldolgozása, ...)

# Összefoglalás

- Elméleti eredményeim:
  - Teljesen igény szerinti Hierarchikus A\* változat kidolgozása
- Gyakorlati eredményeim:
  - Architektúra stabilizáció
  - Teljesen igény szerinti változat **nyílt forráskódú** implementációja a **Theta** modellellenőrző keretrendszerben
  - Teljesítmény kiértékelése verifikációk futtatásával

