# Hibatűrő elosztott rendszerek paraméterezett modellellenőrzése

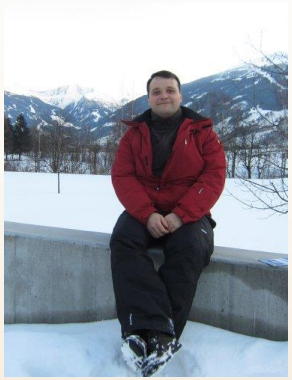
Molnár Vince

**Parameterized model checking of fault-tolerant distributed algorithms by abstraction**

Annu John, Igor Konnov, Ulrich Schmid, Helmut Veith, Josef Widder
*FMCAD*, pages 201-209, 2013.

# A szerzőkről

Annu John



Igor Konnov

Ulrich Schmid

Helmut Veith

Josef Widder

for(syte) Formal Methods in Systems Engineering

# Hibatűrő elosztott rendszerek

- A megbízhatóság növelésére replikálhatjuk rendszereinket
- A fürt elemeinek konzisztens állapotban kell maradniuk
  - Kívülről „egynek látszanak”
  - Hibák jelenlétében is!
- A konzisztencia alapja a konszenzus
  - Minden (hibátlan) példány értsen egyet
  - Nehézség: csak lokális állapot hozzáférhető
- Modell:
  - *n* példányból *f* hibás
  - *t* a hibák maximális száma

# Hibamodellek

- Leállás (fail-stop): jól detektálható leállás
- Összeomlás (crash): leállás hibajelzés nélkül
- Kihagyás (omission): működés vagy üzenet elvesztése
- Időzítési hiba: működés/üzenet „rosszkor”
- Adathiba: hibás adat jelenik meg
- **Bizánci hiba: tetszőleges, félrevezető hiba**
  - A folyamat tetszőlegesen viselkedhet
  - Pl. különböző üzenetek multicastban

# Elméleti eredmények

- (A teljesség igénye nélkül…)
- Aszinkron kommunikáció:
  - Lehetetlen a konszenzus crash típusú hibák esetén (Fischer et al., 1985)
- Szinkron kommunikáció:
  - Lehetetlen a konszenzus köralapú rendszerekben, ha az üzenetek legalább fele elveszhet körönként (Santoro & Widmayer, 1989)
  - Lehetséges a konszenzus bizánci típusú hibák esetén, ha $n > 3t$ (Lamport et al., 1982)

# Példa konszenzus protokollra (Srikanth & Toueg, 87)

```
1.  //Az i. folyamatpéldány változói
2.  v_i : { 0, 1 } initially 0 or 1
3.  accept_i : { 0, 1 } initially 0

4.  //A protokoll egy lépése
5.  if v_i == 1 then send (echo) to all;

6.  if received (echo) from at least
          t + 1 distinct processes
          and not sent (echo) before
7.  then send (echo) to all;

8.  if received (echo) from at least
          n - t distinct processes
9.  then accept_i := 1;
```

# Konzisztencia követelményei (bináris konszenzus)

- Feladat: adott $v_i$ kezdeti érték mellett mindegyik folyamatnak **visszavonhatatlanul** döntenie kell egy közös érték mellett.

- **Egyezség (agreement):** minden hibátlan példány ugyanarra a döntésre jut.
- **Érvényesség (validity):** Ha mindegyik példány $v_i$ értéke megegyezik, akkor a döntés is ez az érték lesz.
- **Végesség (termination):** Minden hibátlan folyamatpéldány előbb-utóbb döntésre jut.

# Követelmények a példa protokollal szemben

- Konszenzus helyett megbízható broadcast
  - Így **aszinkron megbízható üzenetküldéssel** is megoldható
  - Különbség: csak az egyik érték esetén van teendő
- Követelmények:
  - **Hamisíthatatlanság:** ha mindegyik hibátlan folyamatra $v_i = 0$, akkor minden hibátlan folyamat $accept_j$ értéke is 0 marad.
  - **Teljesség:** ha minden hibátlan folyamatra $v_i = 1$, akkor előbb-utóbb az egyik hibátlan folyamatra $accept_j$ értéke 1 lesz.
  - **Terjedés:** ha egy hibátlan folyamat $accept_i$ értékét 1-re állította, akkor előbb-utóbb minden más hibátlan folyamat is így tesz.

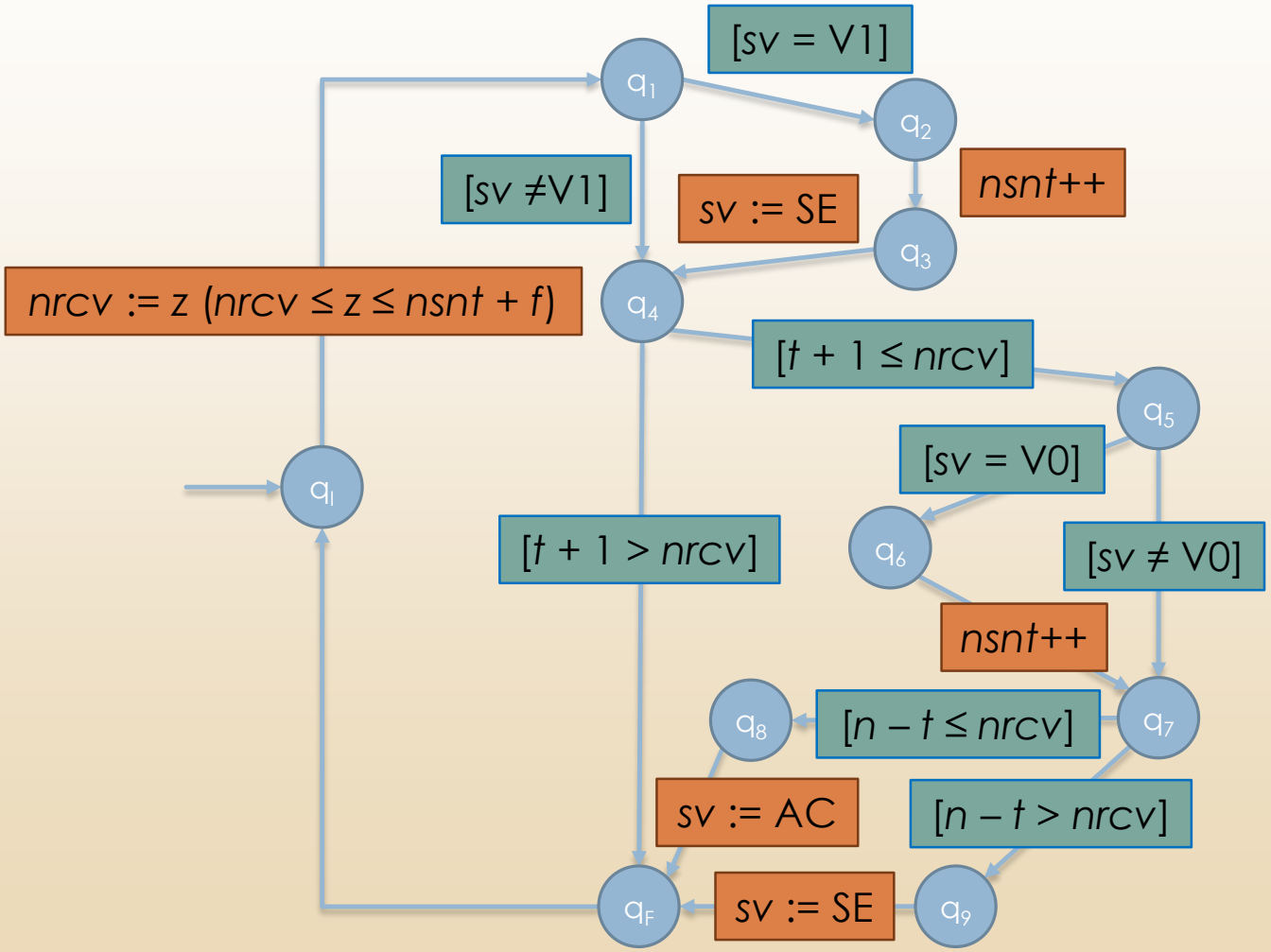
# Paraméterezett modellellenőrzés

- Bemenet:
  - Paraméterezhető folyamatsablon(ok)
  - Paraméterekre vonatkozó kényszerek
  - Követelmények
- Feladat:
  - Teljesülnek-e a követelmények
  - **A paraméterek tetszőleges értéke mellett**
- Visszavezethető rá a megállási probléma
  - Általános esetben nem eldönthető

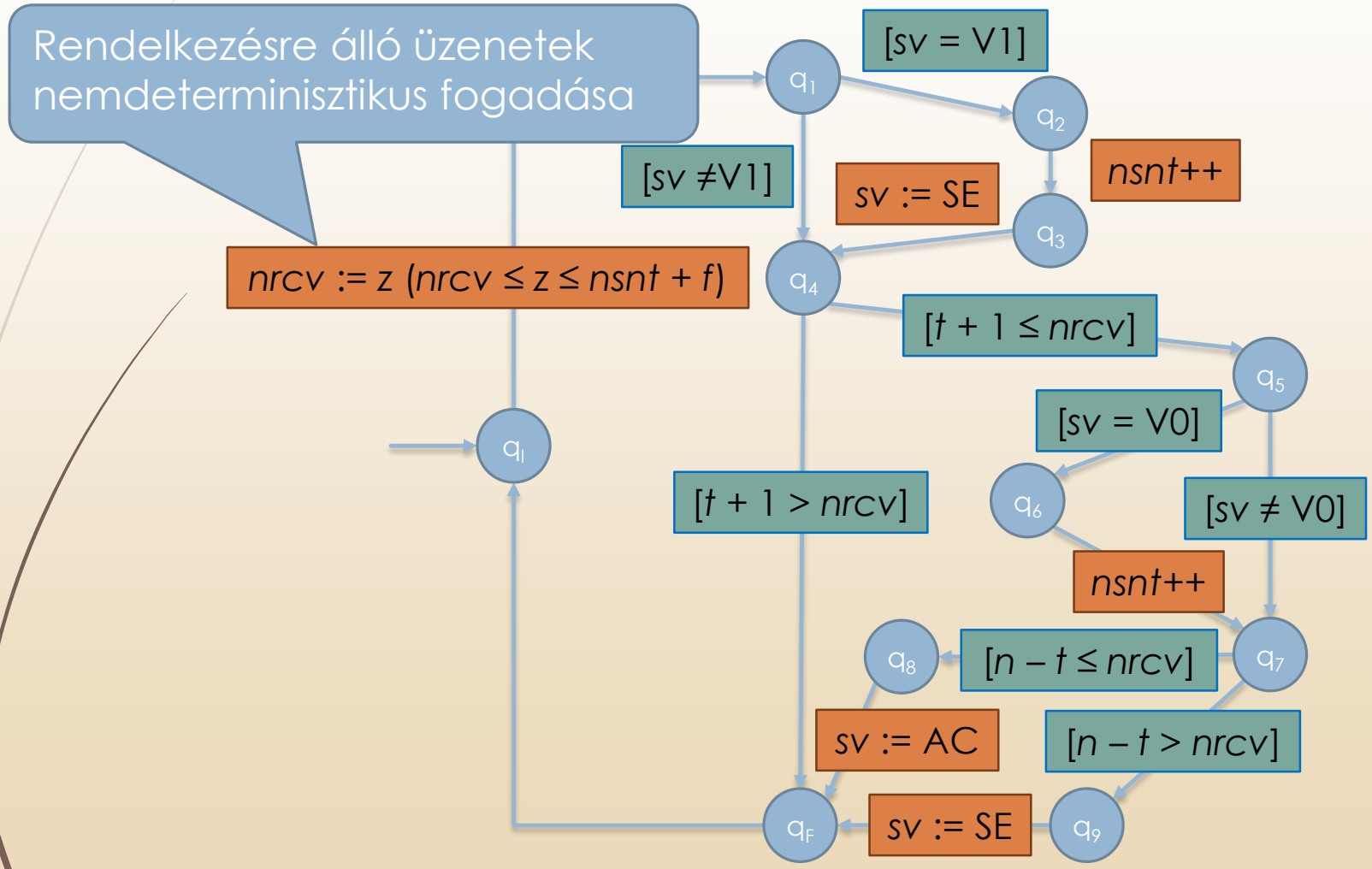
# Üzenetküldés és hibák modellezése

- Változók a küldött és fogadott üzenetek számára:
  - Hibátlan folyamatok által elküldött (globális): *nsnt*
  - Hibátlan folyamatokhoz érkezett (lokális): $nrcv_i$
- Megbízható aszinkron üzenetküldés:
  - *nsnt*++;
- Üzenet fogadás (nemdeterminisztikus átmenet):
  - $[nrcv_i < nsnt] \rightarrow nrcv_i{+}{+}$
- Hibák mellett:
  - $[nrcv_i < nsnt + \mathbf{f}] \rightarrow nrcv_i{+}{+}$
- Fairness követelmény a megbízható üzenetküldésre:
  - $\mathbf{FG}(\forall i.nrcv \geq nsnt)$

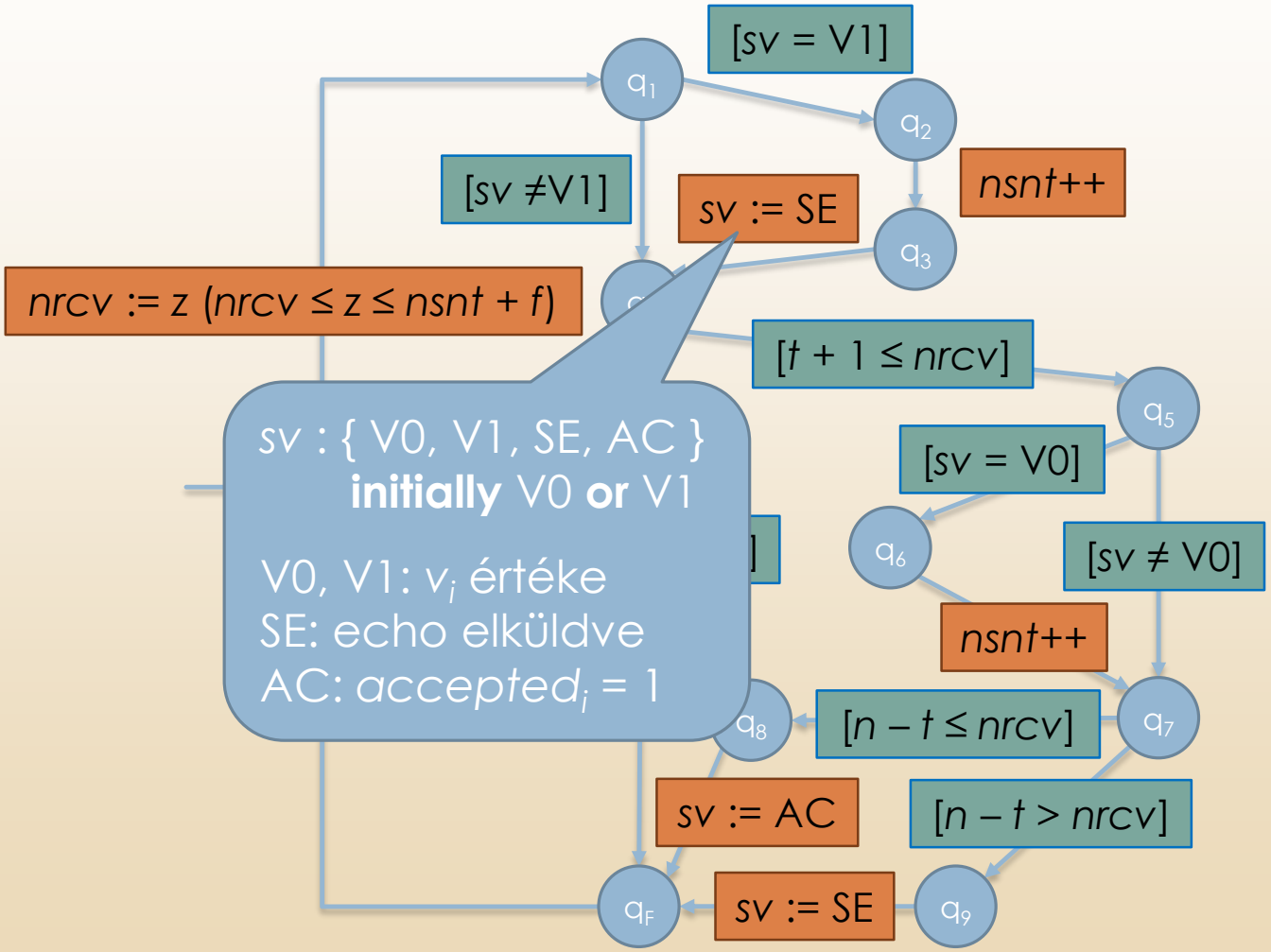
# Control flow automa



# Control flow automa



# Control flow automa



# Adatabsztrakció

- Az $nrcv_i$ konkrét értéke nem érdekes
- Absztrakció négy tartományba:
  - $I_0 = [0, 1)$
  - $I_1 = [1, t + 1)$
  - $I_{t+1} = [t + 1, n - t)$
  - $I_{n-t} = [n - t, \infty)$
- **Paraméterezett Intervallum Absztrakció (PIA)**
- Teljes sorrendezés biztosított a protokoll feltételezése mellett: ***n > 3t***

# Adatabsztrakció

- Feltételek:
  - A teljes sorrendezés miatt intervallumokkal is működik
  - Pl. $nrcv_i \geq t + 1 \rightarrow nrcv_i = I_{t+1} \vee nrcv_i = I_{n-t}$
- Inkrementálás ($x' = x + 1$):
  - $x = I_0 \wedge x' = I_1 \vee$
    $x = I_1 \wedge (x' = I_1 \vee x' = I_{t+1}) \vee$
    $x = I_{t+1} \wedge (x' = I_{t+1} \vee x' = I_{n-t}) \vee$
    $x = I_{n-t} \wedge x' = I_{n-t}$
  - Az absztrakt rendszerben az inkrementálás megtarthatja a korábbi értéket!

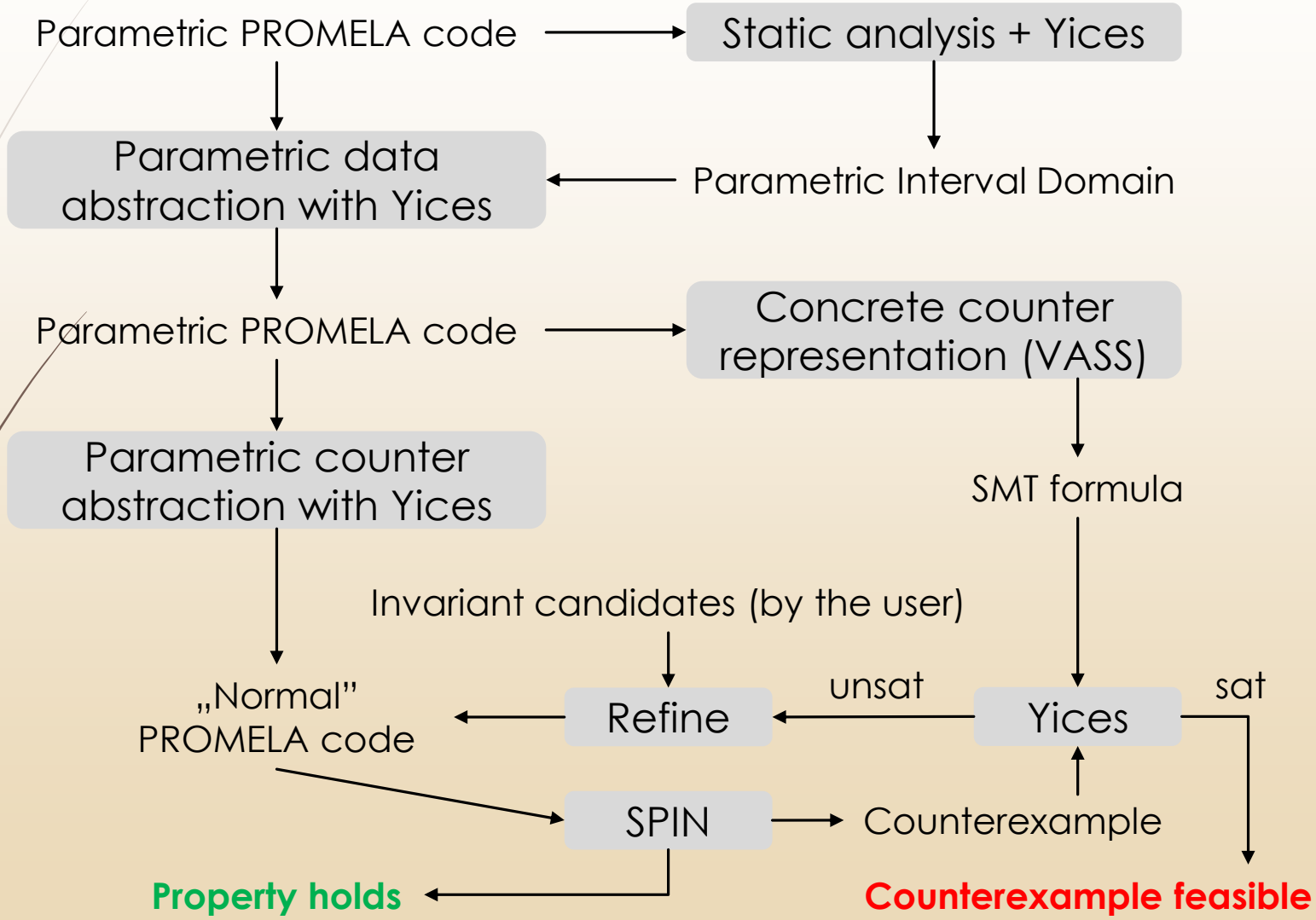
# Counter absztrakció

- A folyamatok lokális állapota az (*sv*, *nrcv*) változópár
- **Counter absztrakció: hány folyamat van egy adott állapotban?**
  - A control flow automata alapján származtatható egy új állapotgép
  - Minden állapothoz tartozik egy változó, ezek értékkészlete [0, *n*]
  - Az előbb definiált paraméteres tartományokkal a változók száma 16-ra csökkenthető
  - A változók értékkészletére is alkalmazható a PIA
- A feltételek továbbra is kiértékelhetők!

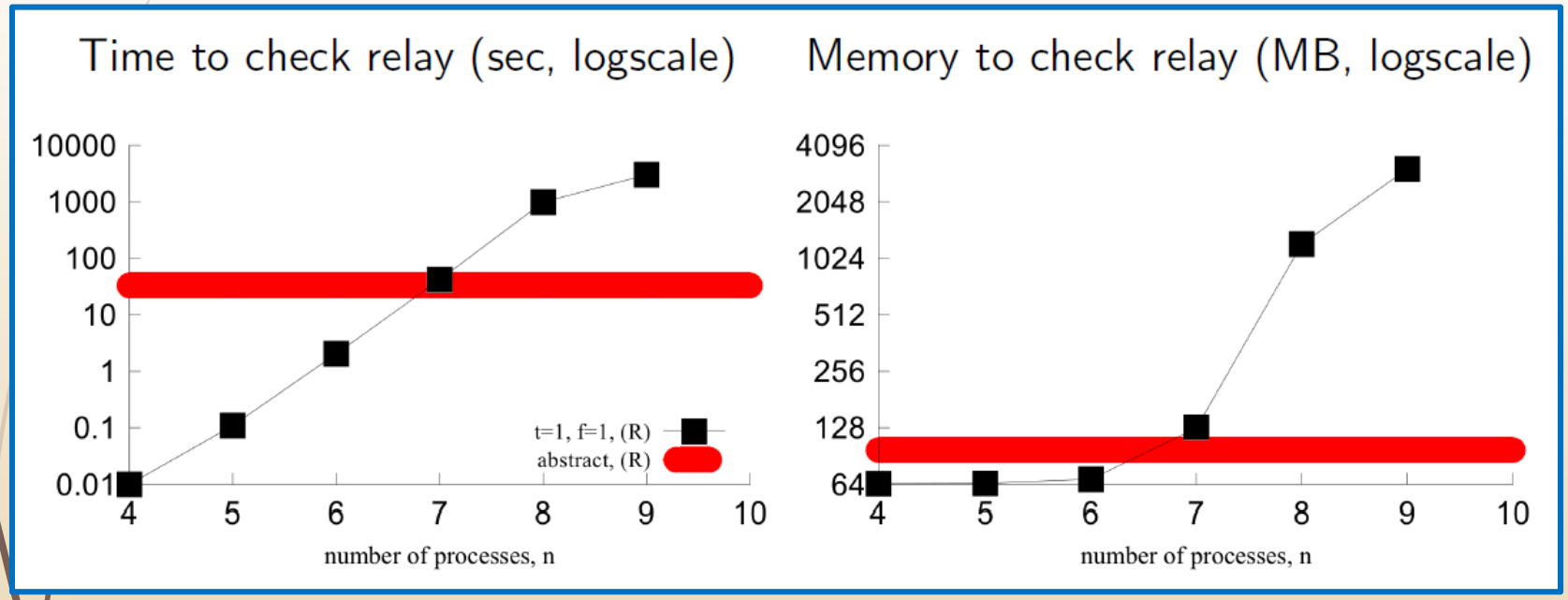
# Modellellenőrzés

- **Eredmény: véges, absztrakt állapotgép**
  - Leírja a protokoll működését minden lehetséges paraméterezés mellett
- Modellellenőrzés segítségével ellenőrizhetők a megadott követelmények
  - DE: az absztrakció miatt hamis ellenpéldák adódhatnak
- CEGAR (**C**ounter**E**xample **G**uided **A**bstraction **R**efinement)
  - Hibás vagy nem elérhető állapotok kivétele a modellből
  - Nem fair vagy bejárhatatlan ellenpéldák eltávolítása a temporális logikai követelmények kiegészítésével

# Tool chain: ByMC



# Mérési eredmények*




*Helmut Veith VTSA előadásából

# Mérési eredmények*

| Algorithm | Fault | Resilience | Property | Valid? | #Refinements | Time |
|---|---|---|---|---|---|---|
| **ST87** | Byz | $n > 3t$ | **U** | ✓ | 0 | 4 sec. |
| **ST87** | Byz | $n > 3t$ | **C** | ✓ | 10 | 32 sec. |
| **ST87** | Byz | $n > 3t$ | **R** | ✓ | 10 | 24 sec. |
| **ST87** | Symm | $n > 2t$ | **U** | ✓ | 0 | 1 sec. |
| **ST87** | Symm | $n > 2t$ | **C** | ✓ | 2 | 3 sec. |
| **ST87** | Symm | $n > 2t$ | **R** | ✓ | 12 | 16 sec. |
| **ST87** | Omit | $n > 2t$ | **U** | ✓ | 0 | 1 sec. |
| **ST87** | Omit | $n > 2t$ | **C** | ✓ | 5 | 6 sec. |
| **ST87** | Omit | $n > 2t$ | **R** | ✓ | 5 | 10 sec. |
| **ST87** | Clean | $n > t$ | **U** | ✓ | 0 | 2 sec. |
| **ST87** | Clean | $n > t$ | **C** | ✓ | 4 | 8 sec. |
| **ST87** | Clean | $n > t$ | **R** | ✓ | 13 | 31 sec. |
| **CT96** | Clean | $n > t$ | **U** | ✓ | 0 | 1 sec. |
| **CT96** | Clean | $n > t$ | **A** | ✓ | 0 | 1 sec. |
| **CT96** | Clean | $n > t$ | **R** | ✓ | 0 | 1 sec. |
| **CT96** | Clean | $n > t$ | **C** | ✗ | 0 | 1 sec. |

*Helmut Veith VTSA előadásából

# Mérési eredmények*

| Algorithm | Fault | Resilience | Property | Valid? | #Refinements | Time |
|---|---|---|---|---|---|---|
| **ST87** | Byz | $n > 3t \wedge f \leq t+1$ | U | ✗ | 9 | 56 sec. |
| **ST87** | Byz | $n > 3t \wedge f \leq t+1$ | C | ✗ | 11 | 52 sec. |
| **ST87** | Byz | $n > 3t \wedge f \leq t+1$ | R | ✗ | 10 | 17 sec. |
| **ST87** | Byz | $n \geq 3t \wedge f \leq t$ | U | ✓ | 0 | 5 sec. |
| **ST87** | Byz | $n \geq 3t \wedge f \leq t$ | C | ✓ | 9 | 32 sec. |
| **ST87** | Byz | $n \geq 3t \wedge f \leq t$ | R | ✗ | 30 | 78 sec. |
| **ST87** | Symm | $n > 2t \wedge f \leq t+1$ | U | ✗ | 0 | 2 sec. |
| **ST87** | Symm | $n > 2t \wedge f \leq t+1$ | C | ✗ | 2 | 4 sec. |
| **ST87** | Symm | $n > 2t \wedge f \leq t+1$ | R | ✓ | 8 | 12 sec. |
| **ST87** | Omit | $n \geq 2t \wedge f \leq t$ | U | ✓ | 0 | 1 sec. |
| **ST87** | Omit | $n \geq 2t \wedge f \leq t$ | C | ✗ | 0 | 2 sec. |
| **ST87** | Omit | $n \geq 2t \wedge f \leq t$ | R | ✗ | 0 | 2 sec. |

*Helmut Veith VTSA előadásából

# Összefoglalás

- Hibatűrő elosztott protokollok
- Paraméterezett modellellenőrzése
- Paraméterezett intervallum absztrakcióval
  - Adaton
  - Állapoton (counter absztrakció)
- Egy egész protokollcsalád kezelése

Felhasznált irodalom:

**Parameterized model checking of fault-tolerant distributed algorithms by abstraction**

Annu John, Igor Konnov, Ulrich Schmid, Helmut Veith, Josef Widder
*FMCAD*, pages 201-209, 2013.