

Generovaný C kód s dokázanými vlastnostmi definovanými v lineární temporální logice

Marek Paška

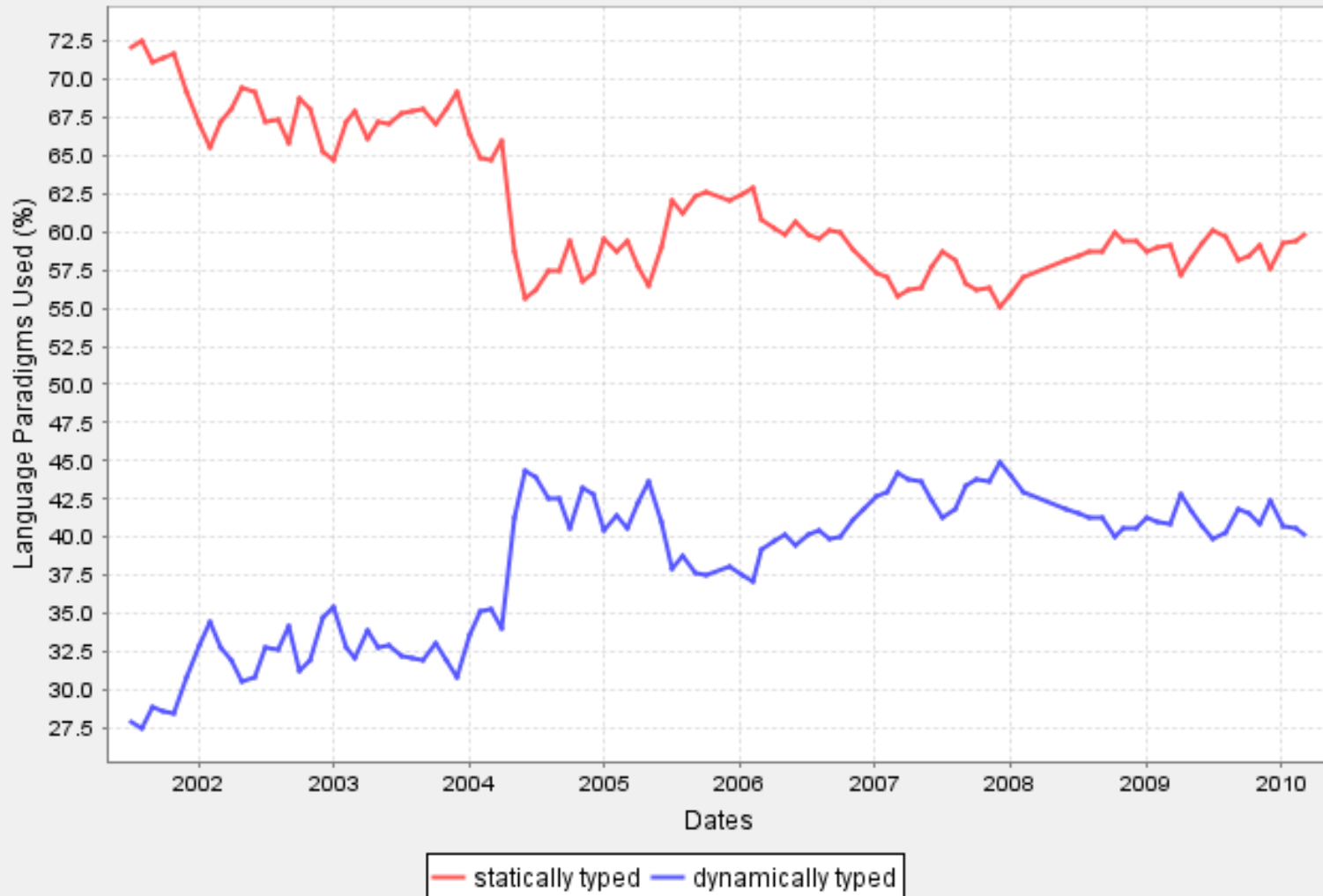
Software ve vestavěných zař.

- omezené zdroje (tlak na cenu zařízení)
- spolehlivý
 - selhání může mít vážné následky
 - chyba se těžko opravuje
- reaguje na podněty z okolí
 - reaktivní
 - reálný čas
- čas programátora levnější než strojový čas

Java + 1 = Python



Tiobe Programming Paradigm Index: Type system

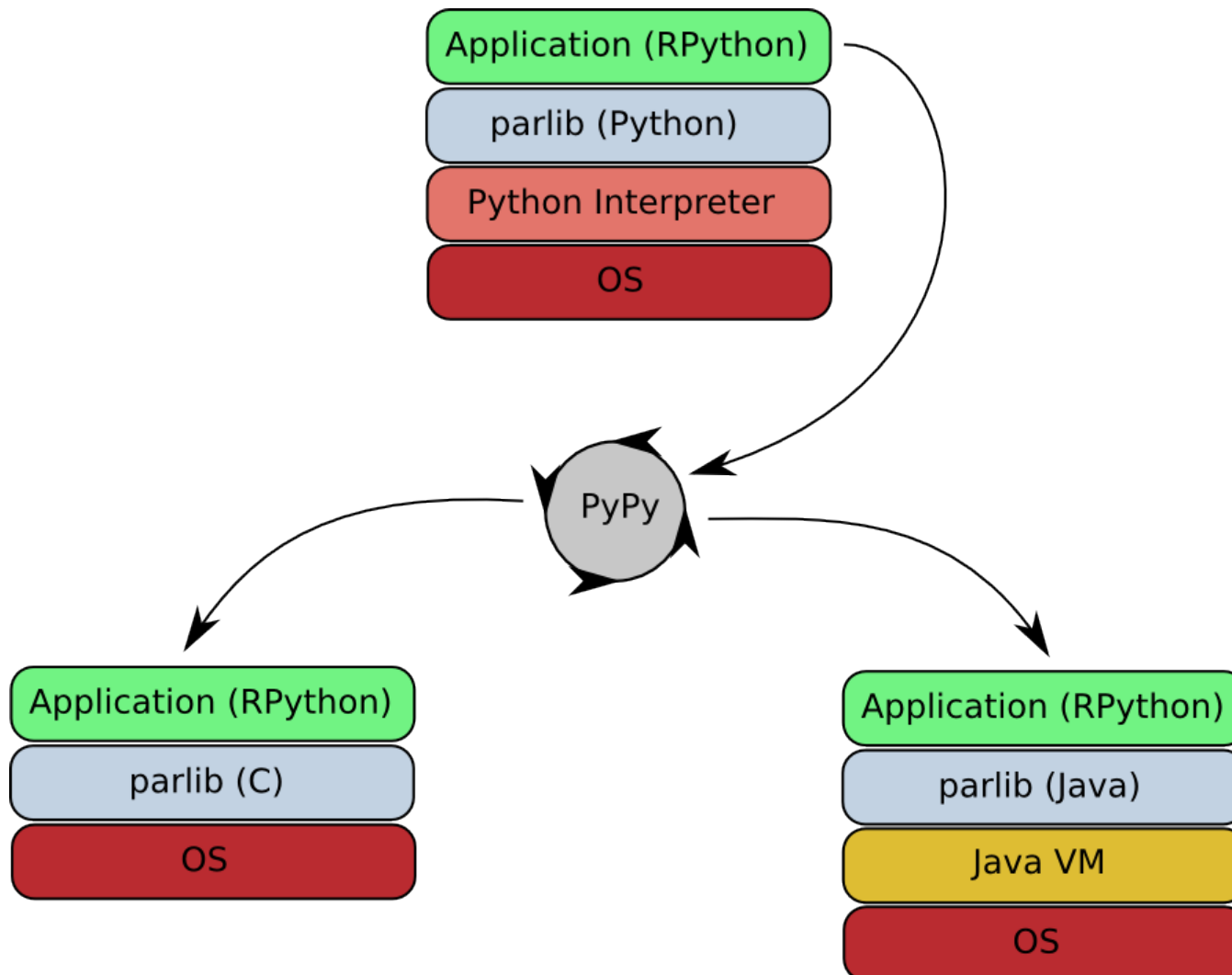


RPython



- podmnožina jazyka Python
 - příjemné programování
- výsledek projektu PyPy (ETH Zürich)
 - experimentální interpret a překladač Pythonu
- dobré vlastnosti dynamicky typovaných jazyků
 - krátký kód (méně chyb)
 - otevřeno pro nová paradigmatata (DbC, AOP)
- překlad do různých jiných kódů (C, JVM)

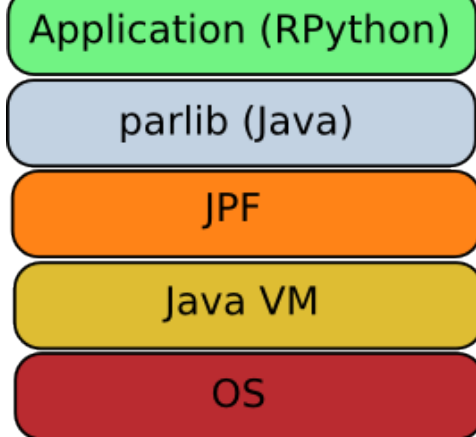
Schéma generování



Java PathFinder



- explicitní model-checker pro Java bytecode
- JVM s backtrackingem
 - uváznutí (deadlock)
 - neodchycené výjimky
 - Linear Temporal Logic (plugin)
- spíše hodně důkladné testování než formální důkaz správnosti



Lineární temporální logika

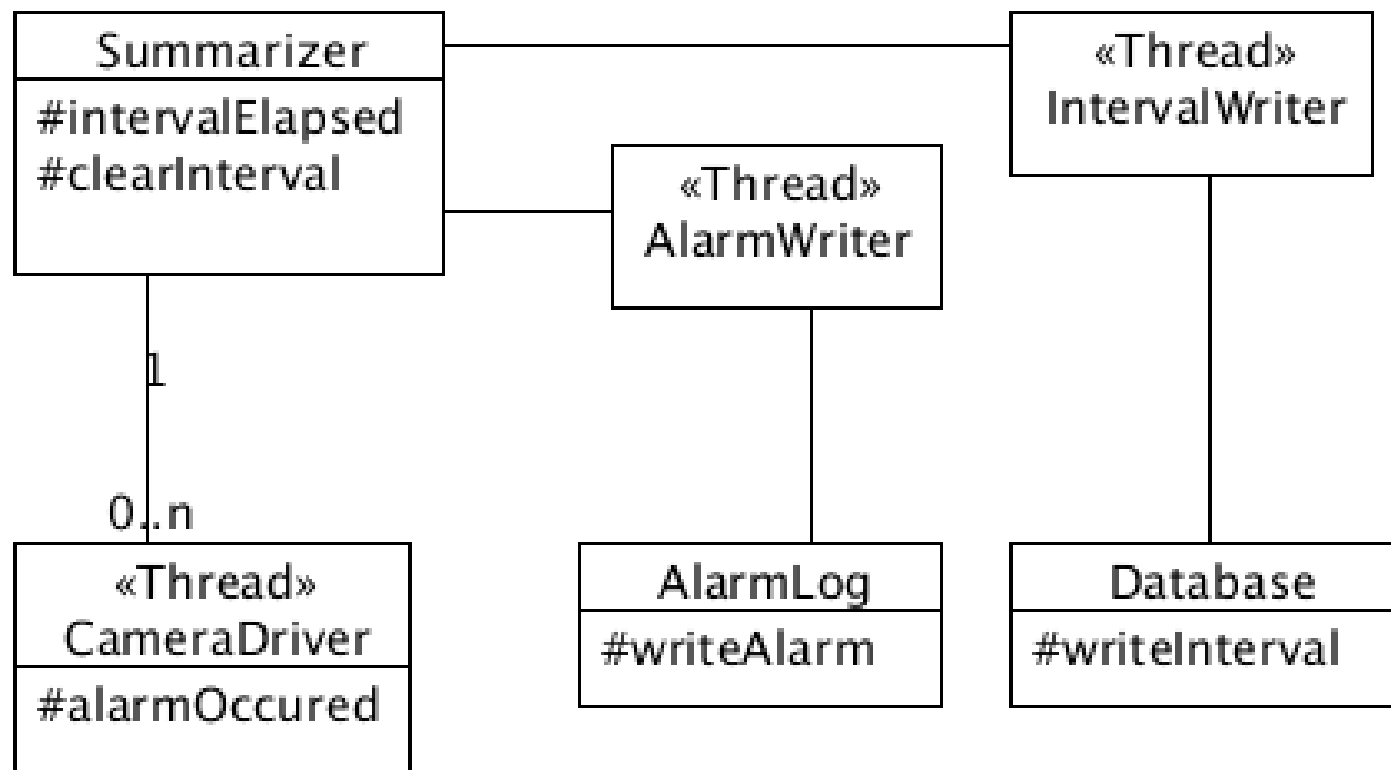
- definována nad sekvencí stavů
 - $s_0 s_1 s_2 s_3 \dots$
- nad každým stavem definována tvrzení
 - $\varphi = x > 4$
 - $\varphi(s_2) = \text{true}$ nebo *false*
- temporální operátory
 - $X\varphi, G\varphi, F\varphi, \varphi_1 U \varphi_2$

LTL - Příklady



- $F(\text{všechny_záznamy_zpracovány})$
 - garantován nějaká pozitivní událost
- $G(\text{aspoň_jedno_vlákno_není_blokováno})$
 - není deadlock
- $G(\text{request} \Rightarrow X(F(\text{response})))$
 - dotaz nevyhnutelně způsobí odpověď
- $G(\neg \text{soubor_uzavřen} \cup \text{výsledek_zapsán})$
 - správná práce se soubory
- $G(F(\text{kladný_zůstatek}))$
 - dobrý hospodář se opakovaně dostává do plusu

Studie: Logovač událostí



LTL formulka pro intervaly

```
G ( (method:Summarizer.intervalElapsed)
    => (X (
        (~ (method:Summarizer.clearInterval))
        U (method:Database.writeInterval)
    )
)
)
```

LTL formulka pro alarmy



```
G ( (method:Driver.alarmOccurred)
      -> (X (F (method:AlarmLog.writeAlarm) ) ) )
```

K verifikaci



- Jak můžu usuzovat, že vlastnosti dokázané pomocí JPF platí i pro C kód?
 - v RPythonu (a tedy v C a Javě) používám javovské synchronizační primitiva: *monitory*
 - C i Java byte-code jsou generovány ze stejného mezikódu, C má navíc dvě transformace (pro výjimky a GC), které jsem zdokumentoval

Konec



- děkuji za pozornost