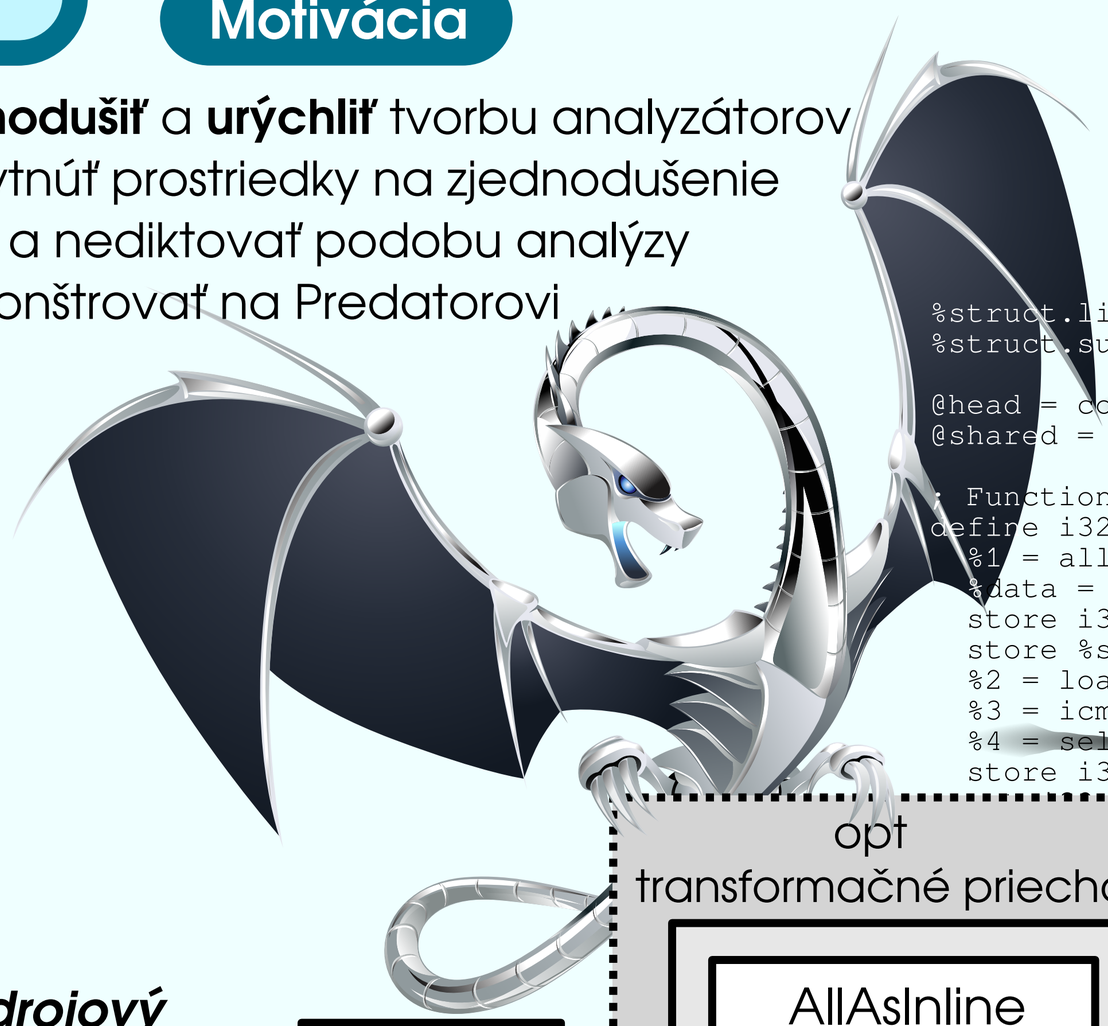
Prostredie pre vývoj statických analyzátorov nad LLVM

12

Motivácia

- **Zjednodušiť** a **urýchliť** tvorbu analyzátorov
- Poskytnúť prostriedky na zjednodušenie kódu a nediktovať podobu analýzy
- Demonštrovať na Predatorovi

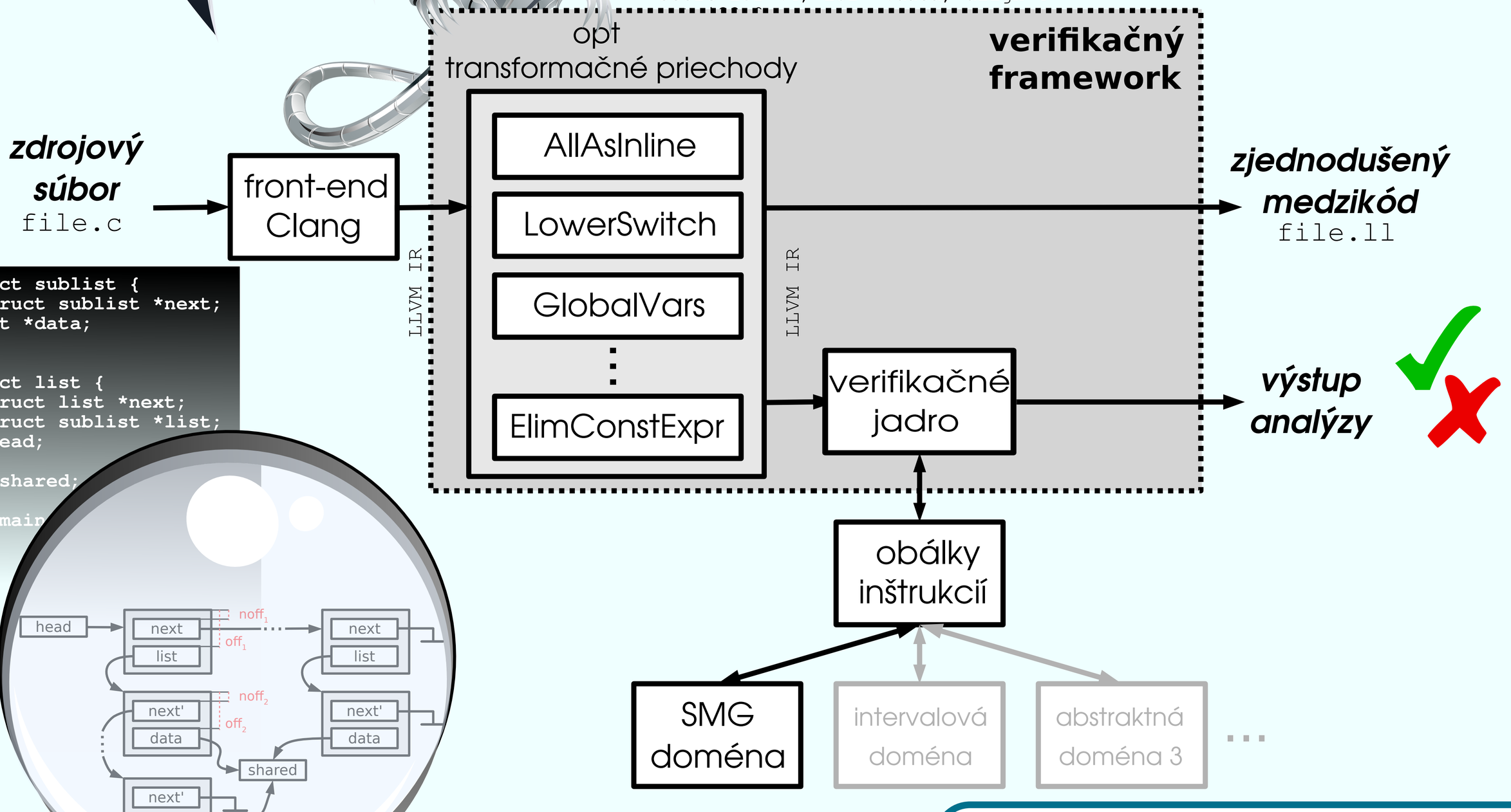


```
%struct.list = type { %st
%struct.sublist = type { %

@head = common global %st
@shared = common global i

; Function Attrs: nounwin
define i32 @main() #0 {
  %1 = alloca i32, align
  %data = alloca i32, alig
  store i32 0, i32* %1, a
  store %struct.list* null
  %2 = load i32, i32* @sha
  %3 = icmp ne i32 %2, 0
  %4 = select i1 %3, i32 1
  store i32 %4, i32* %data, align 4
```

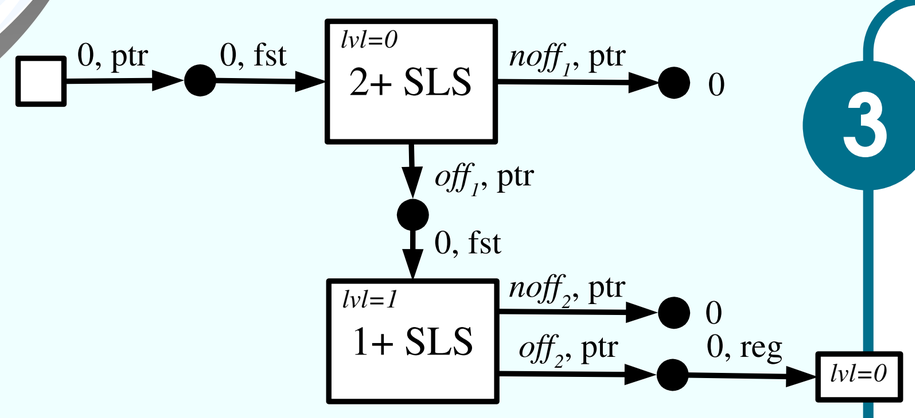
2 Transformačné priechody

Účel: Redukcia množiny konštrukcií jazyka, ktorú musí analyzátor podporovať.

- Sploštenie kódu
- Odstránenie konštantných výrazov
- Odstránenie zložených inicializácií globálnych premenných
- Nahradenie `memcpy` / `memset`
- Odstránenie inštrukcií `switch`, `select`, `phi`




```
struct sublist {
  struct sublist *next;
  int *data;
};

struct list {
  struct list *next;
  struct sublist *list;
} *head;

int shared;

int main
```

1 Vstupné programy Predatora

C programy pracujúce s **dynamickými dátovými štruktúrami** — viacnásobne zanorené jedno- / obojsmerne viazané **zoznamy**. Kontroluje sa práca s pamäťou: neplatné dereferencie, nedefinované smerníky, práca s nízkoúrovňovými operáciami...

3 Verifikácia pomocou symbolických grafov pamäte

- Reimplementácia nástroja **Predator**
- Prehľadávanie stavového priestoru (BFS/DFS) prechádzaním grafu riadenia toku programu
- Symbolické vykonávanie inštrukcií
- **Abstrakcia** neobmedzených stavov pamäte pomocou konečnej reprezentácie **SMG** — graf s objektami (abstraktné reprezentujú segmenty zoznamu) a hodnotami

Veronika Šoková
xsokov00@stud.fit.vutbr.cz

Excel@FIT 2016