Оглавление

Глава 1

Анализ типов

1.1 Что будет, если в нашу систему типов ввести тип $Bool$?

Продублируем изначальные правила:

I	$[[I]] = int$
$E_1 == E_2$	$[[E_1]] == [[E_2]] \wedge [[E_1\ op\ E_2]] = int$
$E_1 op E_2$	$[[E_1]] == [[E_2]] == [[E_1\ op\ E_2]] = int$
$input$	$[[input]] = int$
$X = E$	$[[X]] = [[E]]$
$output\ E$	$[[E]] = int$
$if(E)\ \{S_1\}$	$[[E]] = int$
$if(E)\ \{S_1\}\ else\ \{S_2\}$	$[[E]] = int$
$while\ (E)\ \{S\}$	$[[E]] = int$
$f(X_1, ..., X_n)\ \{...return\ E;\}$	$[[f]] = ([[X_1]], ..., [[X_n]]) \to [[E]]$
$(E)\ (E_1, ..., E_n)$	$[[E]] = ([[E_1]], ..., [[E_n]]) \to [[(E)(E_1, ..., E_n)]]$
$\&E$	$[[\&E]] = \&[[E]]$
$alloc$	$[[alloc]] = \&\alpha$
$null$	$[[null]] = \&\alpha$
$^*E$	$[[E]] = \&[[^*E]]$
$^*X = E$	$[[X]] = \&[[E]]$

Тогда, перво-наперво введём булевый литерал в пару к I - целочисленном литералу:

$$B \Rightarrow [[B]] = boolean$$

Понятно, что возможные значения - это *True* или *False*. Следовательно меняется тип выражений в инструкциях, а также у бинарных операторов - теперь стоило бы выделить логические операторы и арифметические операторы, но т.к. в TIP есть только два логических оператора то нет нужды выписывать какой-нибудь *logOp*.

Ещё одно следствие - мы не знаем тип *input* и *output*. Выпишем изменившиеся правила:

I	$[[I]] = int$
B	$[[B]]] = boolean$
$E_1 \ > \ E_2$	$[[E_1]] == [[E_2]] = int \wedge [[E_1 \ > \ E_2]] = boolean$
$E_1 \ == \ E_2$	$[[E_1]] == [[E_2]] == [[E_1 \ == \ E_2]] = boolean$
$E_1 \ op \ E_2$	$[[E_1]] == [[E_2]] == [[E_1 \ arithOp \ E_2]] = int$
$input$	$[[input]] = \alpha$
$X = E$	$[[X]] = [[E]]$
$output \ E$	$[[E]] = \alpha$
$if(E) \ \{S_1\}$	$[[E]] = boolean$
$if(E) \ \{S_1\} \ else \ \{S_2\}$	$[[E]] = boolean$
$while \ (E) \ \{S\}$	$[[E]] = boolean$
$f(X_1, ..., X_n) \ \{...return \ E;\}$	$[[f]] = ([[X_1]], ..., [[X_n]]) \rightarrow [[E]]$
$(E) \ (E_1, ..., E_n)$	$[[E]] = ([[E_1]], ..., [[E_n]]) \rightarrow [[(E)(E_1, ..., E_n)]]$
$\&E$	$[[\&E]] = \&[[E]]$
$alloc$	$[[alloc]] = \&\alpha$
$null$	$[[null]] = \&\alpha$
$^*E$	$[[E]] = \&[[^*E]]$
$^*X = E$	$[[X]] = \&[[E]]$

1.1.1 Будет ли анализ более полным?

Учитывая, что теперь в инструкциях *if* и *while* условие может быть только типа *Bool*, следовало бы что полнота анализа увеличилась, например можно было бы найти ошибки когда в этих инструкциях условие - это арифметическое выражение, однако семантика языка не совпадает с этим правилом (в обоих инструкциях можно вставить целочисленное значение как условие), поэтому полнота всё-таки упадёт.

1.1.2 Будет ли анализ более точным?

Точность не изменится. Soundness как была такая и осталась.

1.2 Что будет, если в нашу систему типов ввести тип $Array$?

По аналогии введём литера массива, а также пустой массив:

$$\{\ \} \Rightarrow [[\{\ \}]] = [\alpha]$$
$$\{E_1, ..., E_n\} \Rightarrow\ [[E_1]] ==\ ...\ == [[E2]]$$
$$E = \{E_1, ..., E_n\} \Rightarrow\ [[E]] == [\ [[E_1]]\ ]$$

Все элементы массива долнжы иметь один тип, а вообще-то то есть это либо *int* либо *unit* (пустой массив), либо также массив, обозначим тип массива как $[< typename >]$.

1.2.1 Придумайте правила вывода для новых операторов

Введём операцию взятия индекса:

$$E[E_1] \Rightarrow [[E]] == [\alpha]\ \wedge\ [[E_1]] == int\ \wedge\ [[\ E[E_1]\ ]] = \alpha$$
$$E[E_1] = E_2 \Rightarrow [[E]] == [\alpha]\ \wedge\ [[E_1]] == int\ \wedge\ [[\ E[E_1]\ ]] == [[E_2]]\ \wedge\ E_2 = \alpha$$

Индексация происходит по числу, следовательно тип индекса - это *int*, также тип присваемого значения должен соответсвовать типу элемента массива, говоря иначе типу выражения, которое возвращает операция взятия индекса.

1.2.2 Попробуйте протипизировать программу

Используя добавленые правила протипизируем программму:

```
main() {
    var x,y,z,t;
    x = {2,4,8,16,32,64};      [[x]]=[ [[2]] ]=[ int ]

    y = x[x[3]];               [[3]]=int^x=[int]=>[[x[3]]]=int=>[[x[x[3]]]]=int
                                   =>[[y]]=int

    z = {{},x};                [[{}]]=[a]^[[x]]=[int]=>[[z]]= [[int]]

    t = z[1];                  [[z]]=[[int]]^[[1]]=int=>[[t]]=[int]

    t[2] = y;                  [[y]]=int^[[2]]=int^[[t]]=[int]=>[[t[2]]]=int
}
```

В результате получаем:

$$[[x]] = [\ int\ ]$$
$$[[y]] = int$$
$$[[z]] = [\ [\ int\ ]\ ]$$
$$[[t]] = int$$