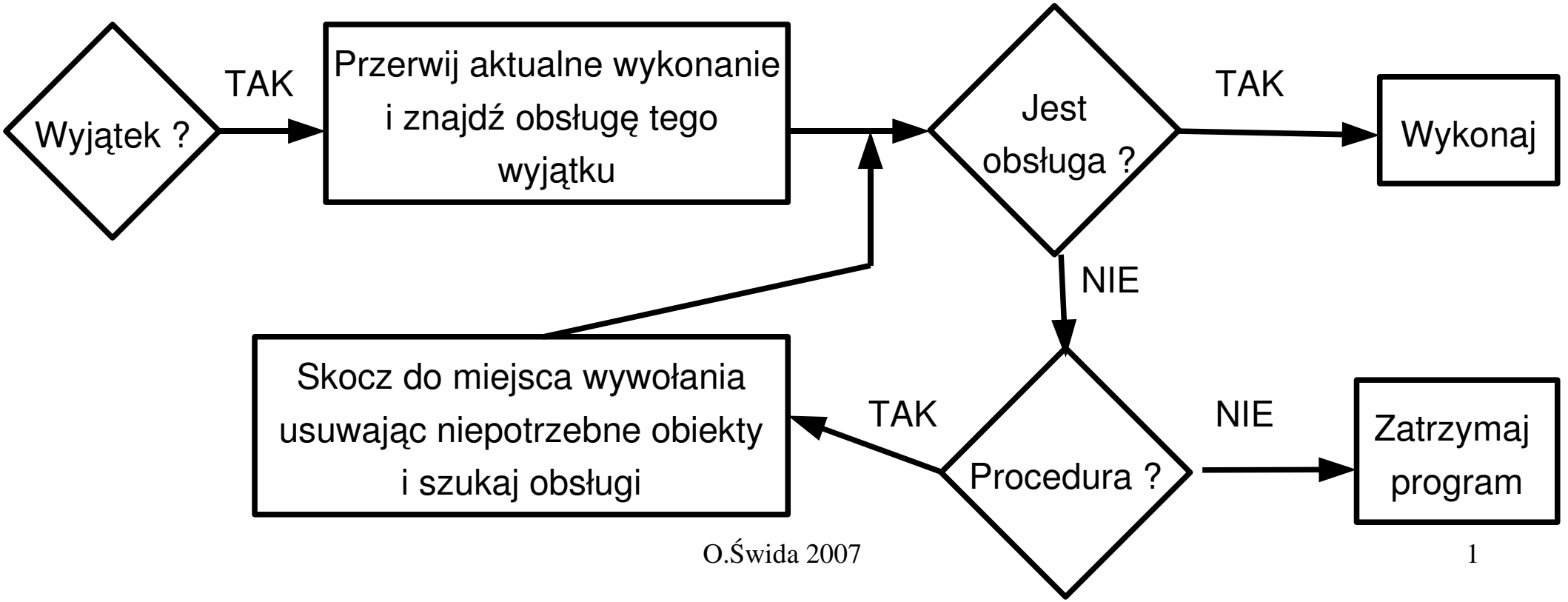
Kompilacja wyjątków (exceptions)

- Pojęcie wyjątku i jego reprezentacja składniowa
- Generacja: jawna (throw) i automatyczna
- Przechwytywanie
- Zwijanie stosu
- Przykład semantyki:



Koncepcja realizacji

- Określ, czy wyjątek nastąpił w kontrolowanym bloku
- Identyfikuj bloki odnoszące się do aktualnego miejsca wystąpienia
- Sprawdź typ wyjątku i określ grupę bloków do jakich pasuje

Jeżeli znaleziono blok – wykonaj jego kod, w przeciwnym wypadku przekaż sterowanie dalej (lub zakończ program)

- *Można wykorzystać mapę bloków obsługi (zakres instrukcji, bloki) oraz mapę typów wyjątków (blok, typ, adres początku instrukcji bloku)*

```
void x() {
    try {   c1;
            y();
            try {   z();
                } catch (e1) { i3;i4;}
        } catch (e) { i1;i2; }
  }
```

Polimorfizm

- Operatory monomorficzne: **boolean a,b;** ⟶ **a** *and* **b**
- Operatory polimorficzne:

  T[] x; int i ⟶ **x[i]**

  T={int,double} T a,b ⟶ **a < b**

- Rodzaje:
  - parametryczny – ten sam algorytm wykonywany dla różnych typów
  - ad-hoc – różne implementacje dla różnych typów (porównanie liczb)
  - podtypów – w językach z koersją (np. porównanie int,float)
- „Polimorficzne” struktury danych – tablice, listy
  - zysk: możliwość sprawdzania typu w czasie kompilacji
- Przykład:

```
class Lista<A> { A dane; Lista(A d) {...} }
int length(Lista<A> lst);
```

Polimorfizm c.d.

- Zdefiniujmy „parametryzowane typy”
  - Lista<int>, Lista<boolean> itp. - otrzymujemy zbiory „możliwych typów”
- Inny pomysł – nie zapisujmy typów wcale! Czy to bezpieczne ?

```
funkcja (x,y) {
    u = x + y;
    v = x – y;
  return u*v;  }
```

```
funkcja2 (a,b,c) {
  if (a) return b;
else return c;
}
```

- Kompilator może automatycznie dodawać informacje o typach (*type inference*)

```
T funkcja2<T> (boolean a,T b,T c) {
  if (a) return b;
  else return c;
}
```

Skąd taki wniosek ?

Polimorfizm c.d.

- Jak wykonywać operacje polimorficzne ?
  - generacja oddzielnego kodu dla każdego użycia polimorficznej funkcji:
    - zwiększony rozmiar programu
    - problemy z kompilacją oddzielnych części programu
  - użycie tego samego kodu dla każdego wywołania
    - mniejszy kod, łatwiejsze w implementacji
    - ale co z rozmiarem danych ?