```
Inductive Pol1 (C : Set) : Set :=
| Pc : C → Pol1 C
| PX : Pol1 C → C → Pol1 C.
```

Pc c : polynoom “c”
PX P a : polynoom “P X + a”

```
Inductive Pol1 (C : Set) : Set :=
| Pc : C → Pol1 C
| PX : Pol1 C → positive → C → Pol1 C.
```

Pc c : polynoom “c”
PX P i a : polynoom “P X^i + a”

```
Inductive Pol1 (C : Set) : Set :=
| Pc : C → Pol1 C
| PX : Pol1 C → positive → C → Pol1 C.

Fixpoint Poln (C : Set) (n : nat) : Set :=
match n with
| O => C
| S m => Pol1 (Poln C m)
end.
```

Pc c : polynoom “c”
PX P i a : polynoom “P X^i + a”

Poln m : type van polynomen in m variabelen

```
Inductive Pol (C : Set) : Set :=
| Pc : C → Pol C
| Pinj : positive → Pol C → Pol C
| PX : Pol1 C → positive → Pol C → Pol1 C.
```

Pc c : “c”
Pinj j Q : “Q{X1 ← X(j+1); …; X(n-j) ← Xn}”
PX P i Q : “P X1^i + Q{X1 ← X2; …; X(n-1) ← Xn}”
(with P != 0)

```
Inductive PolExpr : Set :=
| PEc : C -> PolExpr
| PEX : positive -> PolExpr
| PEadd : PolExpr -> PolExpr -> PolExpr
| PEsub : PolExpr -> PolExpr -> PolExpr
| PEmul : PolExpr -> PolExpr -> PolExpr
| PEopp : PolExpr -> PolExpr.
```