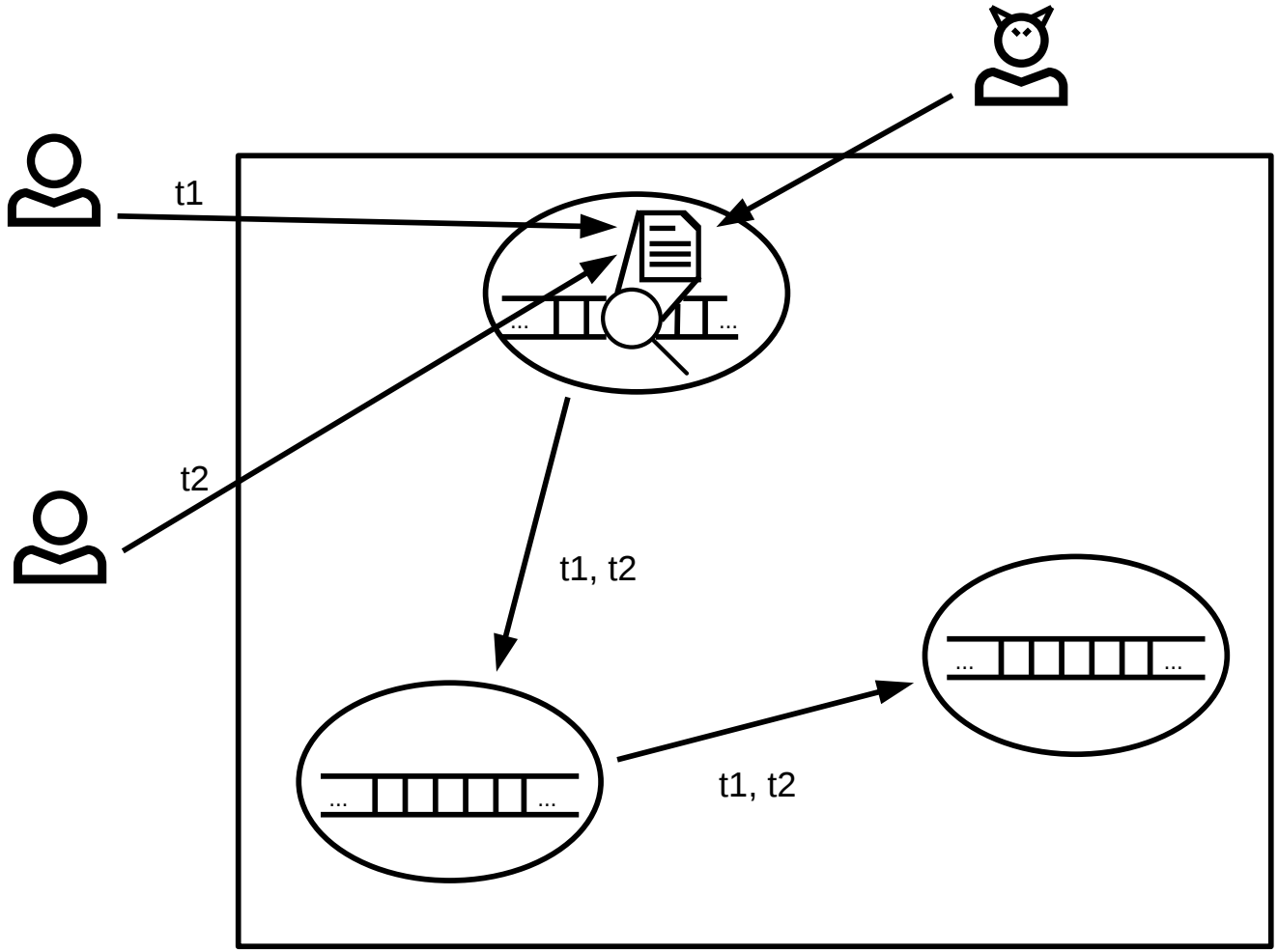
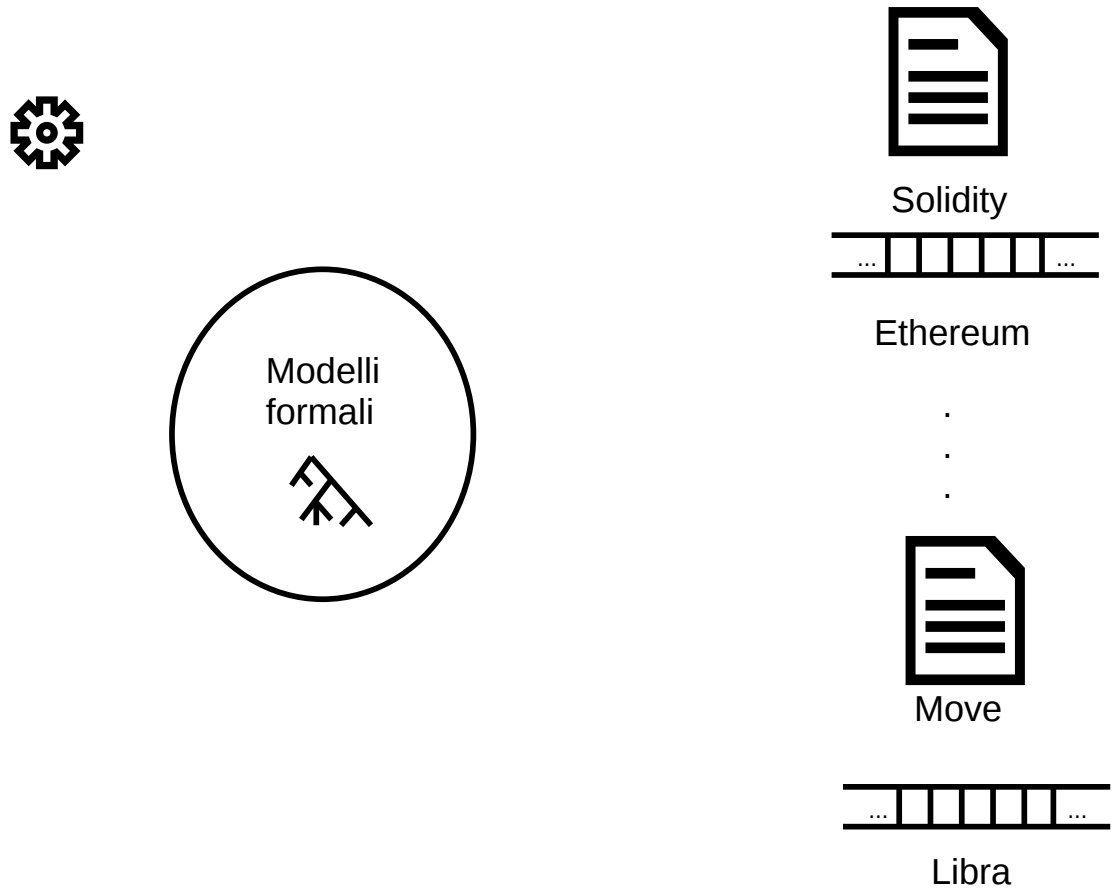
Un compilatore per un linguaggio per smart contract intrinsecamente tipato

Stefano Bucciarelli

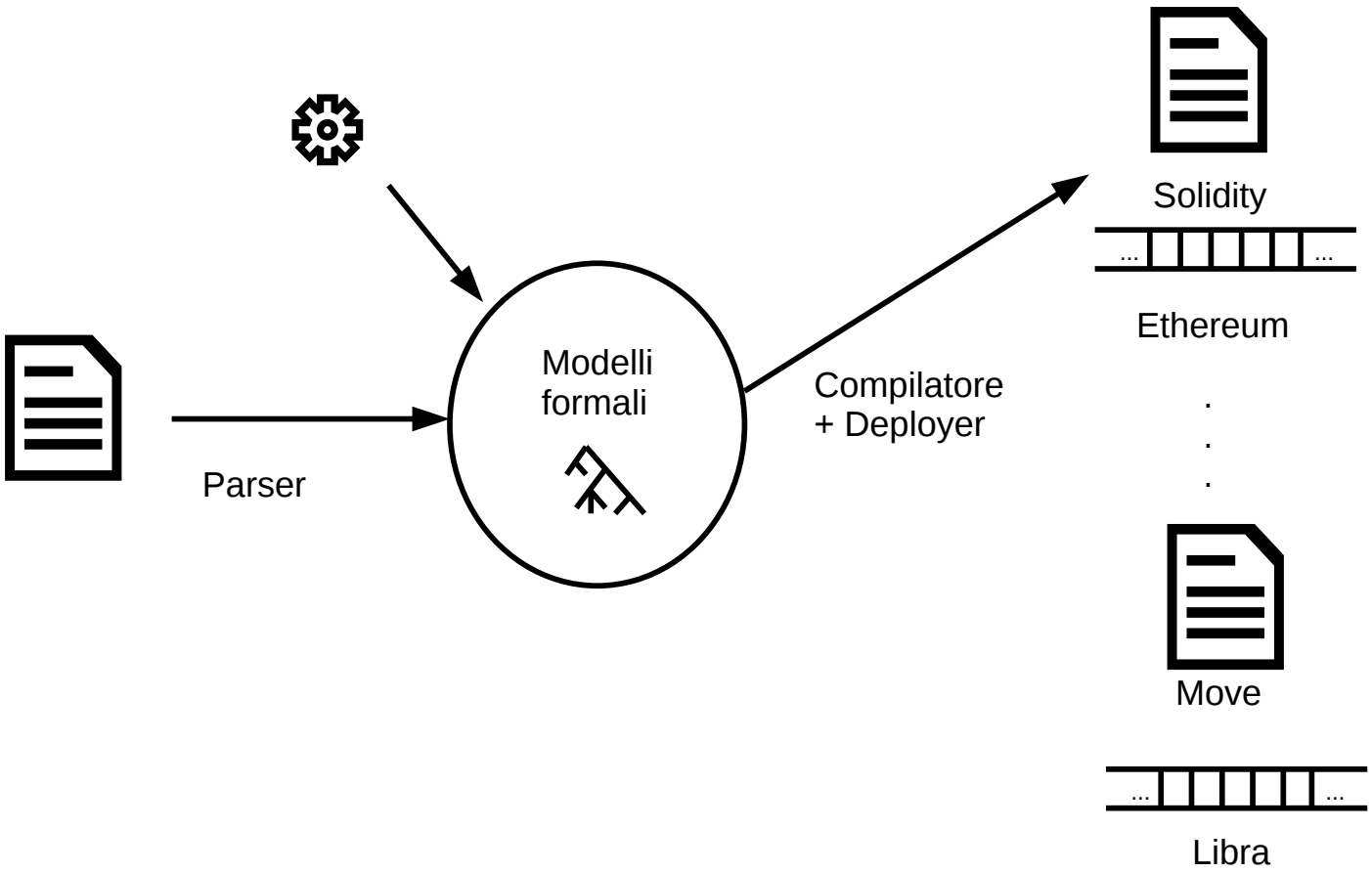
Descrizione smart contract



Analisi su smart contract



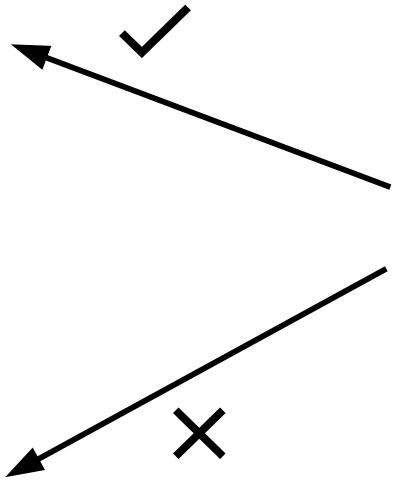
Analisi su smart contract



Intrinsically Typed Data Structure

ADT

```
type expr =
| Int of int
| Bool of bool
| And of expr * expr
| Plus of expr * expr
| Eq of expr * expr
```



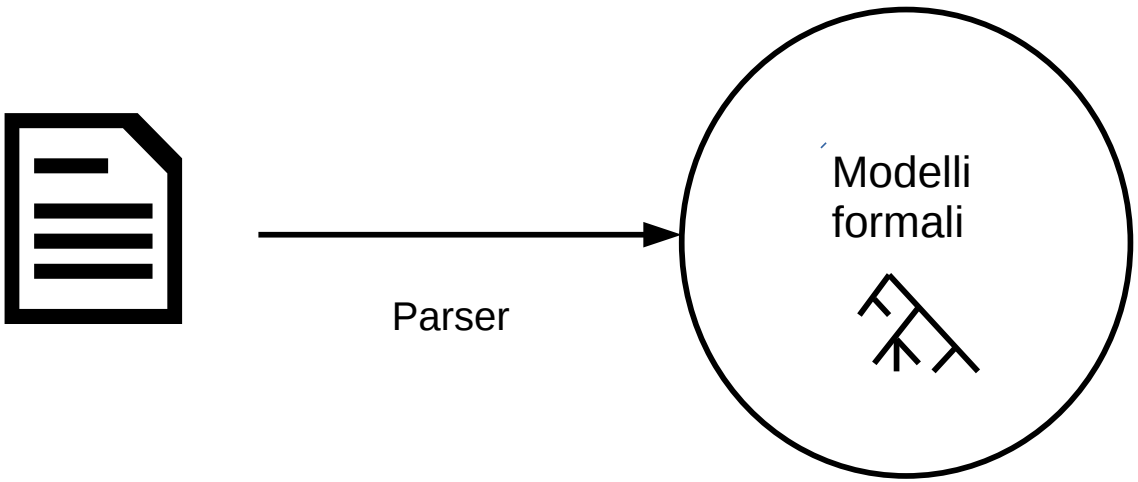
And((Int 9),(Bool true))

GADT

```
type 'a expr =
| Int : int -> int expr
| Bool : bool -> bool expr
| And : bool expr * bool expr -> bool expr
| Plus : int expr * int expr -> int expr
| Eq : 'a expr * 'a expr -> bool expr
```

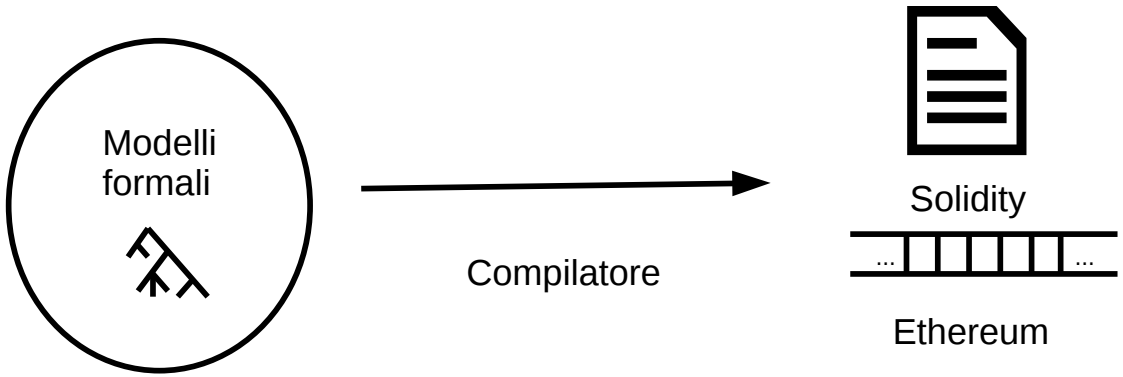
Parsing

- Parsing = Parser + Type checking
- Parser combinator



Compilazione e deploy

- Compilazione tramite la costruzione di un AST intrinsecamente tipato per Solidity
- Inferenza di interfacce
- Script Python per il deploy



Compilazione e deploy

- Compilazione tramite la costruzione di un AST intrinsecamente tipato per Solidity
- Inferenza di interfacce
- Script Python per il deploy

```
Contract sample{
    Contract a
    Contract b
    function foo() {
        a.f(5)
        b.g(6, true)
    }
}
```

Linguaggio sorgente


```
interface Interf0{
    function f(int) external;
}
interface Interf1{
    function g(int, bool) external;
}
contract sample {
    Interf1 b;
    Interf0 a;

    function foo() public{
        a.f(5);
        b.g(6, true);
    }
}
```

Solidity

Lavoro svolto

- Risultati
- 1400 righe di codice OCaml

Futuri lavori

- Codice Python per umani
- Analisi statiche
  - Modelli differenti
  - Modularità