Godot: All the Benefits of Implicit and Explicit Futures

K. Fernandez-Reyes, D. Clarke, L. Henrio, E. B. Johnsen, T. Wrigstad

1 BACKGROUND

CONTROL-FLOW FUTURES (Java)
- Control intermediate async. steps
- Typed futures (Explicit)

DATA-FLOW FUTURES (JavaScript)
- Access data value
- Future type is not visible (Implicit)

2 PROBLEMS

TYPE PROLIFERATION
Types mirror communication structure

FUTURE PROLIFERATION
Depth of n nested futures adds n additional future operations

FULFILMENT OBSERVATION
Impossible to check intermediate future results

3 EXAMPLE

actor PrintServer
    ... 
def print(doc: Doc): Fut[Fut[Bool]]
        this.idleWorker()!print(doc)
    end
end

actor EduPrinter
    def print(doc: Doc): Fut[Bool]
        ...
    end
end

actor PrintServer
    ... 
def print(doc: Doc): Flow[Bool]
        this.idleWorker()!print(doc)
    end
end

actor EduPrinter
    def print(doc: Doc): Flow[Bool]
        ...
    end
end

4 GODOT SYSTEM

CONTROL- & DATA-FLOW EXPLICIT FUTURES
Integration of both futures into single calculus

RESOLVE FUTURE PROBLEMS
- Data-flow futures solve Type Proliferation
- Runtime optimisation solves Future Proliferation
- Control-flow futures solve Fulfilment Observation

Flow types are compressed

\[
\Gamma \vdash e : \tau \\
\Gamma \vdash \rho \text{async}^* e : \downarrow \text{Flow}\tau
\]

Future fulfilment delegation

\[
\text{fresh } j \\
(task_j \ E[\text{return async}^* e]) \rightarrow (task_j e)
\]

\[
\text{Fut } \tau \equiv \square \text{Flow } \tau
\]