

A Formal Analysis Framework for PLEXIL

Gilles Dowek (Ecole Polytechnique, France)

César A. Muñoz (NIA)

Corina Păsăreanu (NASA)

Presented By: Vandi Verma (NASA)

3rd Workshop on Planning and Plan Execution for Real-World Systems Principles and Practices for Planning in Execution

September 2007

Plan Execution Interchange Language (PLEXIL)

- Reactive plan execution language, developed by NASA, to support autonomous operations in a multi-platform environment.
- PLEXIL programs consists of a set of **nodes** that are organized in a tree structure, which provides scope for local variables and basic control structure.
- Node execution is triggered by **gate conditions**, e.g., start, end, and repeat, that react to external and internal events.
- Node execution is monitored by **check conditions**, e.g., precondition, postcondition, and invariant, that signal potentially anomalous behaviors.

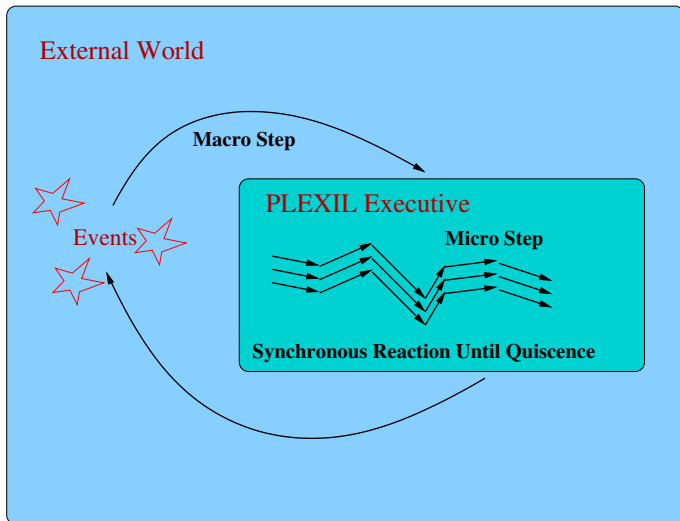
Example

```
Node SafeDrive {
  int pictures=0;
  Repeat-while:
    LookupOnChange(Rover.WheelStuck)==false;
  List: {
    Node OneMeter {
      Command: Rover.Drive(1);
    }
    Node TakePic {
      Start: OneMeter.status==FINISHED AND
            pictures<10;
      Command: Rover.TakePicture();
    }
    Node Counter {
      Start: TakePic.status==FINISHED;
      Pre: pictures<10;
      Assignment: pictures:=pictures+1;
    }
  }
}
```

Formal Semantics vs. Intended Semantics

- **Formal Semantics:** Mathematical description of PLEXIL execution.
- **Intended Semantics:** Description in natural language of PLEXIL executive.
- **Our Objective:** Provide a formal semantics of PLEXIL that
 - ① Identifies *fundamental concepts*, e.g., internal/external states, atomic computations, execution steps, and
 - ② Separate them from *implementation issues*, e.g., efficiency, optimizations, event management.

PLEXIL Execution



Small-Step Semantic Framework

- PLEXIL execution is structured in 5 execution layers:
 - *atomic*: State transitions and memory updates.
 - *micro*: Synchronous execution of the atomic relation.
 - *quiescence*: Run until completion of the micro relation.
 - *macro*: Reaction to events in the external world.
 - *execution*: Iteration of the macro relation.
- This modular approach enables the study of different semantic variants of the language.

Properties

- Determinism.
- Operational determinism.
- Run to completion.
- Stuttering.
- Compositionality.

Fundamental Properties of PLEXIL (I)

- **Determinism:** Given a sequence of readings of the external world, the behavior of a PLEXIL program can be predicted, i.e., the behavior of a PLEXIL program is *reproducible* in a simulation environment.
- **Operational Determinism:** If the changes in the external world do not trigger events in a PLEXIL program, the behavior of the program can be predicted, i.e., the only non-determinism that is allowed by the language comes from uncertainties in the external world.

Fundamental Properties of PLEXIL (II)

- **Run to Completion:** When the quiescence cycle terminates, it reaches a stable state, i.e., if no further event occurs, the internal state of the PLEXIL program remains invariant.
- Corollary:
 - **Stuttering:** If the environment does not change the state of a program does not evolve.

Fundamental Properties of PLEXIL (III)

- **Compositionality**: PLEXIL programs that do not share variables are compositional, i.e., the execution of parallel processes can be inferred from the independent execution of each one of them.

Contributions

- **Formal semantics** of PLEXIL:
 - Prerequisite to theoretical study of the language, and the mathematical understanding the language and its features.
 - Guarantees that PLEXIL execution is well-defined, and that, therefore, PLEXIL programs are (formally) verifiable.
 - Baseline for tools such as the PLEXIL executive and the PLEXIL checker.
- **A modular framework** applicable to other synchronous languages with many semantic variants.
- This framework has been fully developed in the PVS theorem prover and it is freely available as a **PVS library**.

Formal Development in the Program Verification System (PVS)

- 11 Theories (1,500 lines of specification)
Including 1 theory per relation: atomic, micro, quiescence, macro, execution, and 3 theories of abstract relations: reduction, normalized reduction, and synchronous reduction.
- 172 Lemmas (30,000 lines of proofs)
Including 5 main theorems: Determinism, Operational Determinism, Run to Completion, Stuttering, Compositionality
- Main Issues:
 - Construction of modular specifications of the execution relations that support different semantic variations of the language.
 - Finding sufficient conditions to establish theorems such as determinism of synchronous relations and compositionality.

Questions and Comments

Please submit them to the the authors:

- Gilles Dowek (Gilles.Dowek@polytechnique.fr).
- César Muñoz (munoz@nianet.org).
- Corina Păsăreanu (pcorina@email.arc.nasa.gov).

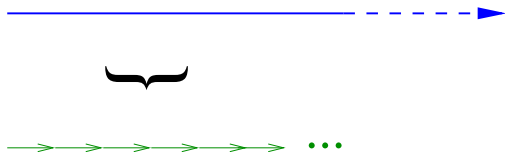
Execution of PLEXIL: Macro Level

Execution defined in terms of external events and interactions with the external world.



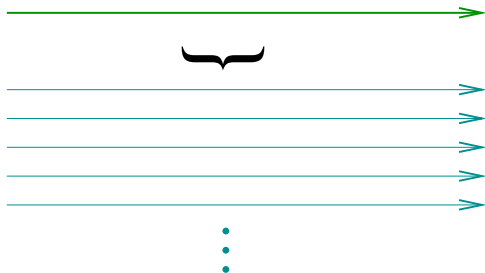
Execution of PLEXIL: Micro Level

Execution defined in terms of internal computations and the local perception of the external world.



Execution of PLEXIL: Atomic Level

Execution defined in terms of parallel state transitions.



PLEXIL Execution: Multi-layer Approach

Execution of PLEXIL is specified by a stack of relations that defines 5 execution layers.

Relation	Symbol	Key Concept
<i>atomic</i>	\longrightarrow	Transition diagrams
<i>micro-step</i>	\Longrightarrow	Parallelism
<i>quiescence</i>	$\Longrightarrow \downarrow$	Quiescence cycle
<i>macro-step</i>	$\star \rightarrow$	Event-driven execution
<i>execution-step</i>	$\star \rightarrow^n$	Full-knowledge