Petri Net Plans

Matteo Leonetti

Dipartimento di Informatica
e Sistemistica Antonio Ruberti

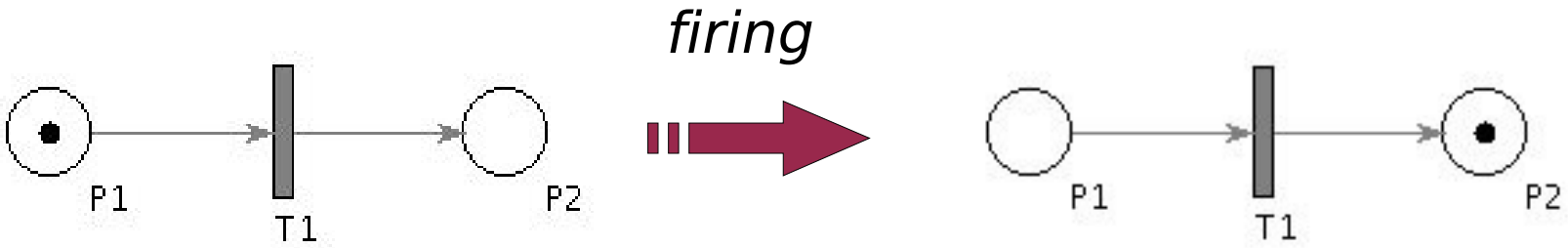
Sapienza
Università di Roma

Outline

- Petri Net Plans
  - Petri Net
  - Motivation and Features
  - Plans
- PNP library
  - A Thousand “Hello World”!

Petri Nets

- Petri Nets are graphs
  - Nodes can be p*laces* or t*ransitions*
  - The state of the network is called *marking*
- PNs are more compact than Finite State Automata (in their general formulation PNs have strictly more expressive power than FSAs)



Let's consider this question

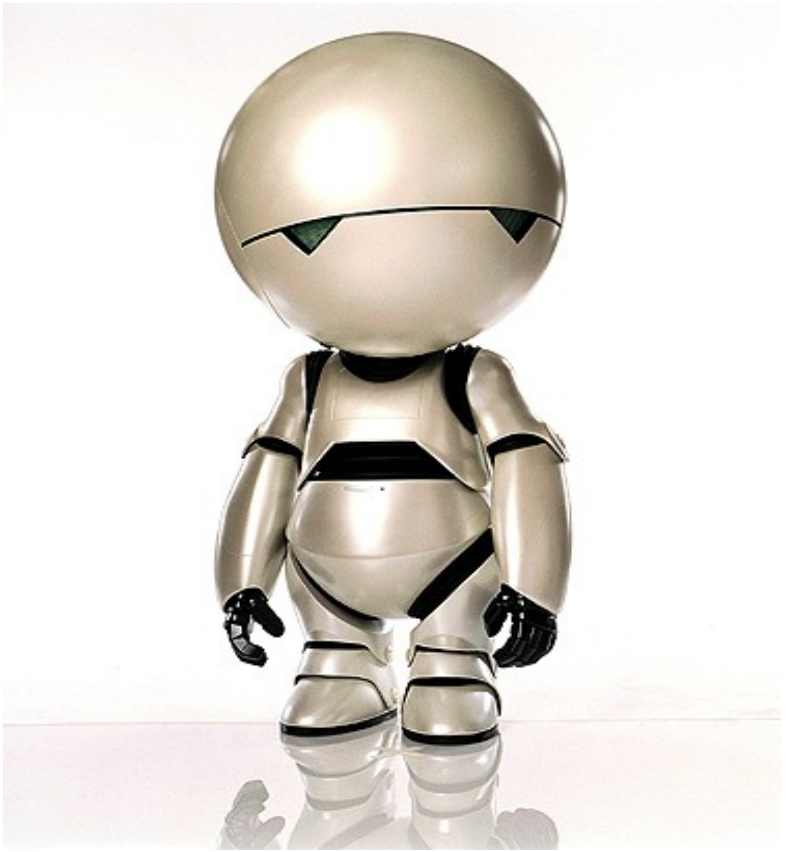
Why do we need Petri Nets?

Classical Planning

Suppose you have...

Classical Planning

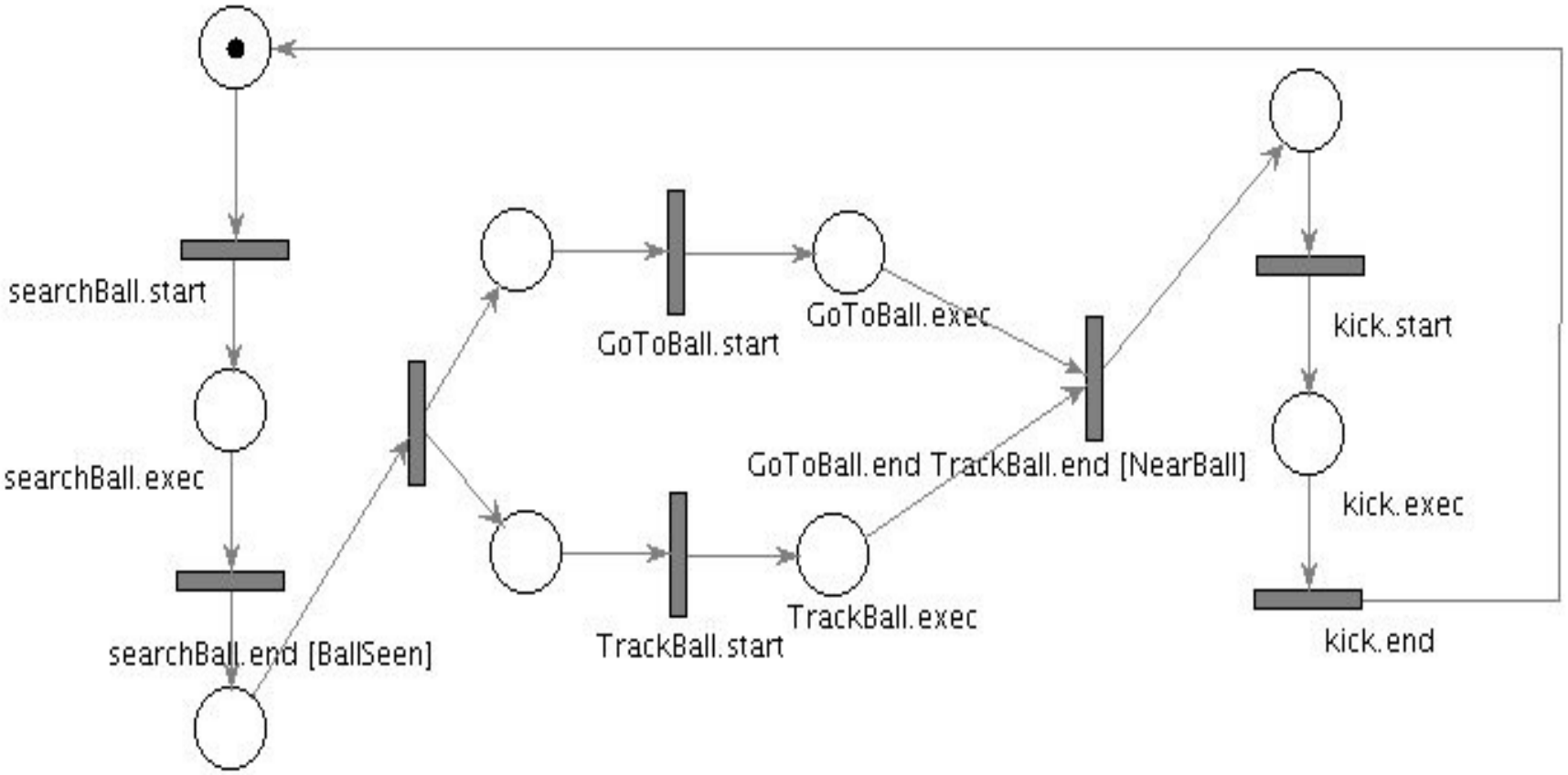
… a **Row Boat**



Planning in the Real World

- Non Instantaneous Actions
- Partial knowledge
- Exogenous Events
- Parallel Execution
- Abstraction

Petri Net Plans



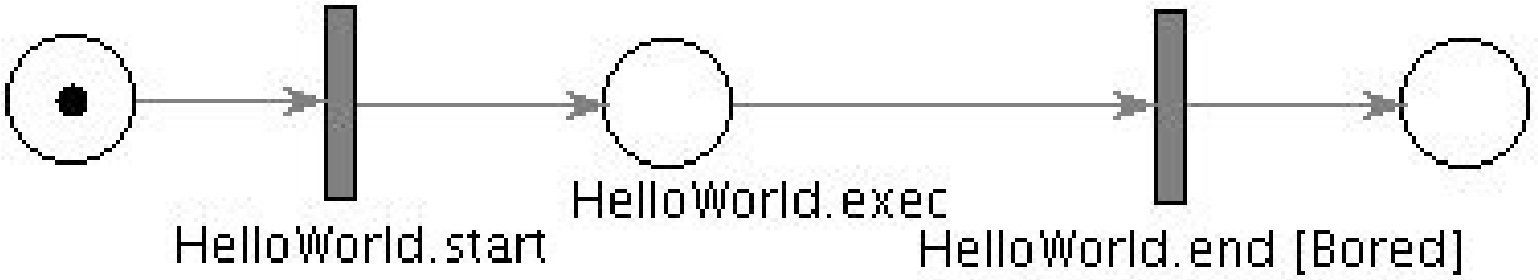
PNP Library

- Petri Net Plans
  - Petri Net
  - Motivation
  - Plans
- PNP library
  - A Thousand “Hello World”!

PNP library

- In order to set up the executor you need to develop:
  - Actions
  - A Condition Checker
  - An Executable Instantiator
  - Plans

Hello World – The Plan



Hello World – The Module

```
class TestPnpModule : public Module {
public:
        TestPnpModule() : executor(NULL) { }
        virtual ~TestPnpModule() { }

        bool initConfigurationProperties();
        // bool initInterfaceProperties();
        bool init();
        void exec();
        // void exitRequested();
        // void cleanup();
        // void asyncAgentCmd(cstr cmd);
        private:
        PetriNetPlans::PnpExecuter<PetriNetPlans::PnpPlan> *executor;
};
```

Hello World – The Module

```
bool TestPnpModule::init()
{
        SimpleInstantiator *simpleIn = new SimpleInstantiator();

        executor = new PnpExecuter<PnpPlan>(simpleIn);
        executor->setMainPlan("hello");
        //...
}
void TestPnpModule::exec()
{
        while (session->wait(), !exiting) {
                SESSION_TRY_START(session)
                        executor->execMainPlanStep();
                SESSION_END_CATCH_TERMINATE(session)
        }
}
```

Hello World – Instantiator

```
class SimpleInstantiator : public PetriNetPlans::ExecutableInstantiator {

public:

PetriNetPlans::PnpExecutable* createExecutable(const std::string& name)
throw();

};
```

Hello World - Instantiator

```
PnpExecutable* SimpleInstantiator::createExecutable(const std::string&
name) throw() {
if(name == "HelloWorld")
        return new HelloWorld();

ExternalConditionChecker *checker = new SimpleConditionChecker();
PnpPlan *plan = new PnpPlan(this,checker);

static const string path("/home/matish/Documents/openRDK-seminari/");
string pnml = path + name +".pnml";

XMLPnpPlanInstantiator planLoader;
```

Hello World - Instantiator

```
try {
        planLoader.loadFromPNML(pnml, plan);
} catch(const runtime_error &) {/*ignore... don't do this at home*/}

return plan;
}
```

Step by step

```
PnpExecutable* SimpleInstantiator::createExecutable(const std::string&
name) throw() {
if(name == "HelloWorld")
        {}//return new HelloWorld();

//ExternalConditionChecker *checker = new SimpleConditionChecker();
PnpPlan *plan = new PnpPlan(this,NULL /*checker*/);

static const string path("/home/matish/Documents/openRDK-seminari/");
string pnml = path + name +".pnml";

XMLPnpPlanInstantiator planLoader;
```

Step by step

```
PnpExecutable* SimpleInstantiator::createExecutable(const std::string&
name) throw() {
if(name == "HelloWorld")
        {}//return new HelloWorld();

ExternalConditionChecker *checker = new SimpleConditionChecker();
PnpPlan *plan = new PnpPlan(this,checker);

static const string path("/home/matish/Documents/openRDK-seminari/");
string pnml = path + name +".pnml";

XMLPnpPlanInstantiator planLoader;
```

Hello World - Conditions

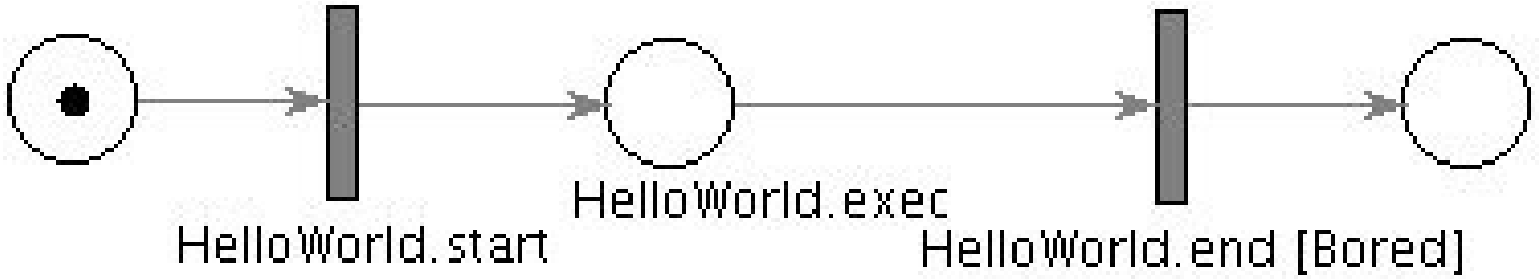
```
class SimpleConditionChecker : public
                PetriNetPlans::ExternalConditionChecker {

bool evaluateAtomicExternalCondition(const std::string& );

};
```

Hello World – The Plan

- What is the parameter of **`evaluateAtomicExternalCondition?`**
- Every condition in the plan. In this example: Bored



```
bool SimpleConditionChecker::evaluateAtomicExternalCondition(const std::string&
condition) {

        if(condition == "Bored") {
                static int times = 0;
                times++;
                times %= 1000;

                return times == 0;
        }

        return true;

}
```

Step by step

```
PnpExecutable* SimpleInstantiator::createExecutable(const std::string&
name) throw() {
if(name == "HelloWorld")
        return new HelloWorld();

ExternalConditionChecker *checker = new SimpleConditionChecker();
PnpPlan *plan = new PnpPlan(this,checker);

static const string path("/home/matish/Documents/openRDK-seminari/");
string pnml = path + name +".pnml";

XMLPnpPlanInstantiator planLoader;
```

Hello World - Action

```
class PnpExecutable {
public:
        PnpExecutable() { }
        virtual ~PnpExecutable() { }
        virtual void start() = 0;
        virtual void resume() = 0;
        virtual void end() = 0;
        virtual void interrupt() = 0;
        virtual void fail() = 0;
        virtual void executeStep() = 0;
        virtual bool finished() = 0;
        virtual bool failed() = 0;
};
```

Hello World - Action

```
class HelloWorld : public PetriNetPlans::PnpAction {

public:

void executeStep();
void end();

};
```

Hello World - Action

```
void HelloWorld::executeStep() {

        std::cout << "Hello World!" << std::endl;

}

void HelloWorld::end() {

        std::cout << "Done" << std::endl;

}
```

PNP library

- In order to set up the executor you need to develop:
  - Actions - **done**
  - A Condition Checker - **done**
  - An Executable Instantiator - **done**
  - Plans – **done**
- **Test it now!**

Conclusion

So long
and thanks
for all the attention