西安电子科技大学
XIDIAN UNIVERSITY

计算机科学与技术学院
SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY
国家示范性软件学院
NATIONAL PILOT SCHOOL OF SOFTWARE ENGINEERING

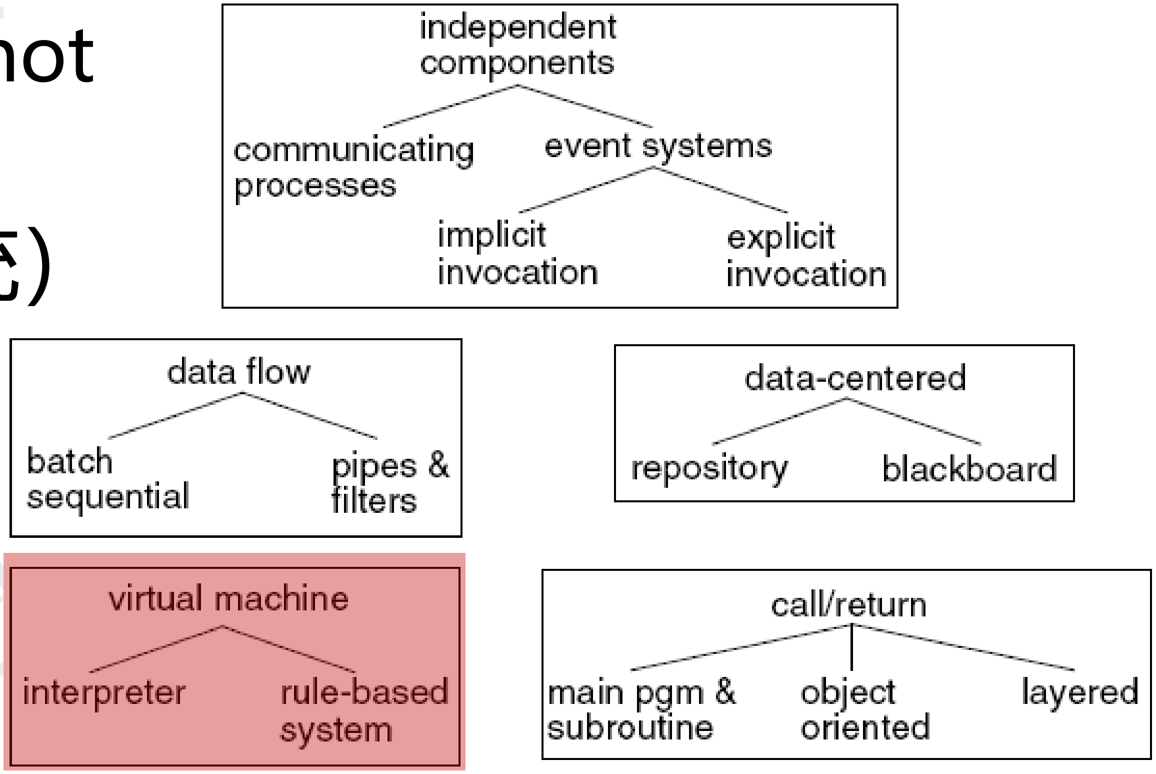
软件体系结构
—虚拟机风格

主讲人：鲍亮 副教授

本节内容

虚拟机风格

- Interpreters（解释器）
  - Simulate functionality which is not native to the hardware
- Rule-based systems（规则系统）
  - Specialization of an interpreter
- Other
  - Syntactic shells
  - Command language processors



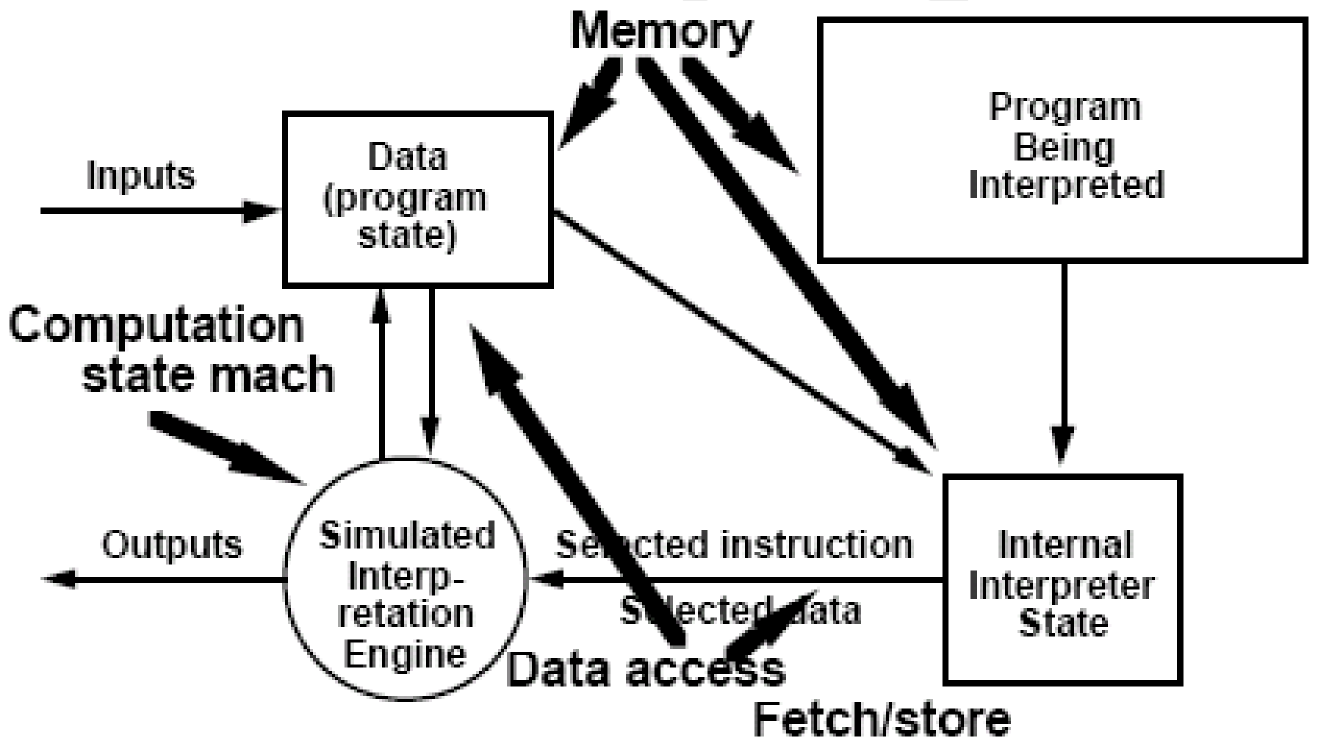
解释器风格

- **Problem:** This pattern is suitable for applications in which the most appropriate language or machine for executing the solution is not directly available. The pattern is also suitable for applications in which the core problem is defining a notation for expressing solutions, for example as scripts. Interpreters are sometimes used in chains, translating from the desired language/machine to an available language/machine in a series of stages.
- **Context:** The interpreter will most often be designed to bridge the gap between the desired machine or language and some (possibly virtual) machine or language already supported by the execution environment.

解释器风格

- **Solution:**
  - *System model:* virtual machine
  - *Components:* one state machine (the execution engine) and three memories (current state of execution engine, program being interpreted, current state of program being interpreted)
  - *Connectors:* data access and procedure call
  - *Control structure:* usually state-transition for execution engine; input driven for selection of what to interpret
- **Significant Variants:** Expert systems are often implemented as interpreters for the collections of rules, or productions, that represent the expertise. Because the productions require a complex selection rule, specialized forms of interpreters have evolved.

解释器风格


Memory
Program Being Interpreted
Inputs
Data (program state)
Computation state mach
Outputs
Simulated Interp-retation Engine
Selected instruction
Selected data
Data access
Internal Interpreter State
Fetch/store

解释器风格：优势

- Functionality:
  - – Can simulate non-native functionality
- Testing:
  - – Can simulate “disaster” modes (e.g. for safety-critical applications)
- Flexibility:
  - – Very general-purpose tool

解释器风格：劣势

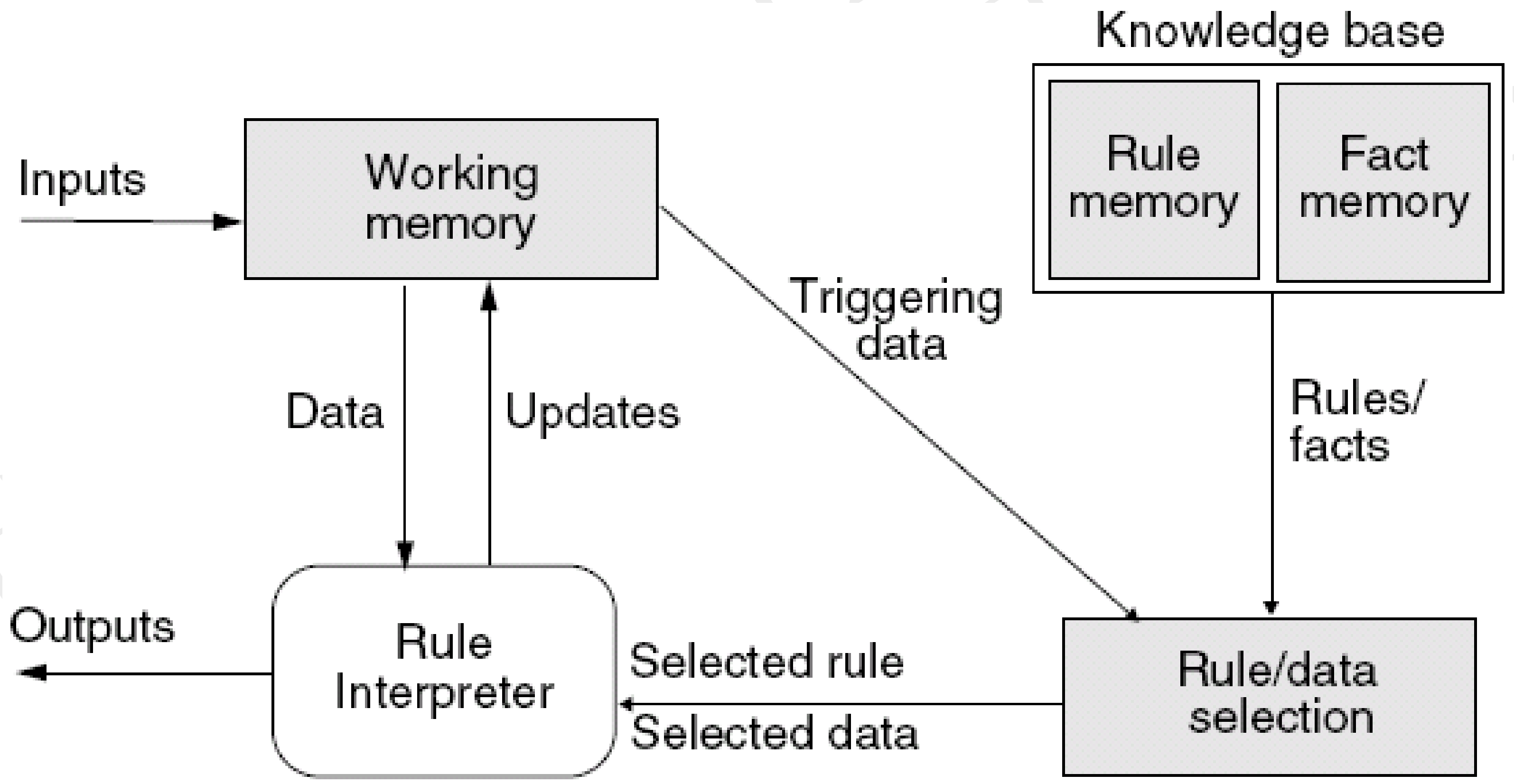
- Efficiency:
  - Much, much slower than hardware
  - Much slower than compiled system
- Testing:
  - Additional layer of software to be verified

A FILM BY CHRISTOPHER NOLAN
INCEPTION
FROM THE DIRECTOR OF THE DARK KNIGHT

解释器风格：用途

- 解释型语言
  - VB、Javascript、VBScript、HTML、Java字节码、Matlab
  - 脚本、配置文件
- 通信协议
- 用户输入
  - 游戏中的组合按键

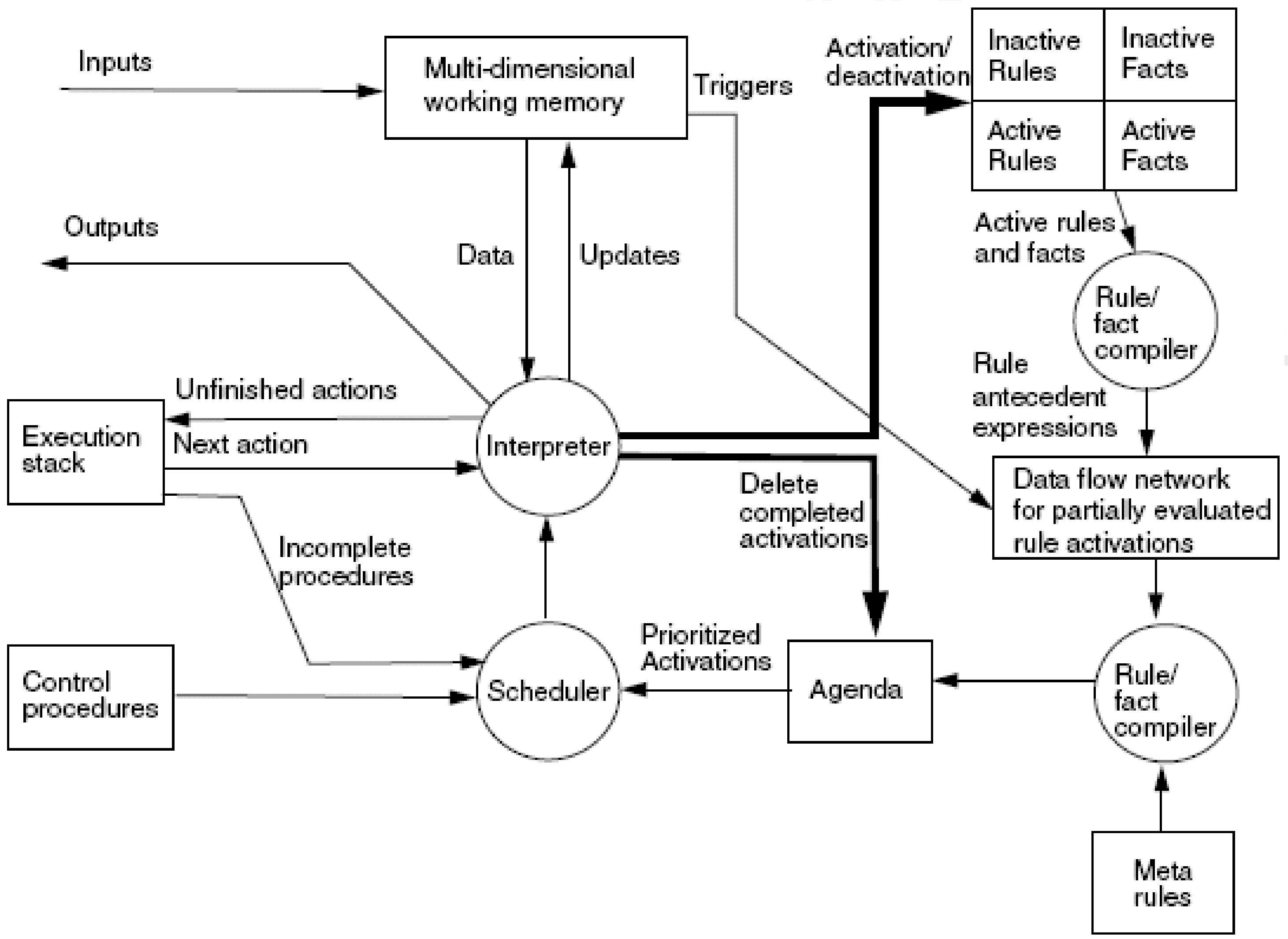
规则系统风格


Knowledge base
Rule memory
Fact memory
Inputs
Working memory
Triggering data
Rules/ facts
Data
Updates
Outputs
Rule Interpreter
Selected rule
Selected data
Rule/data selection

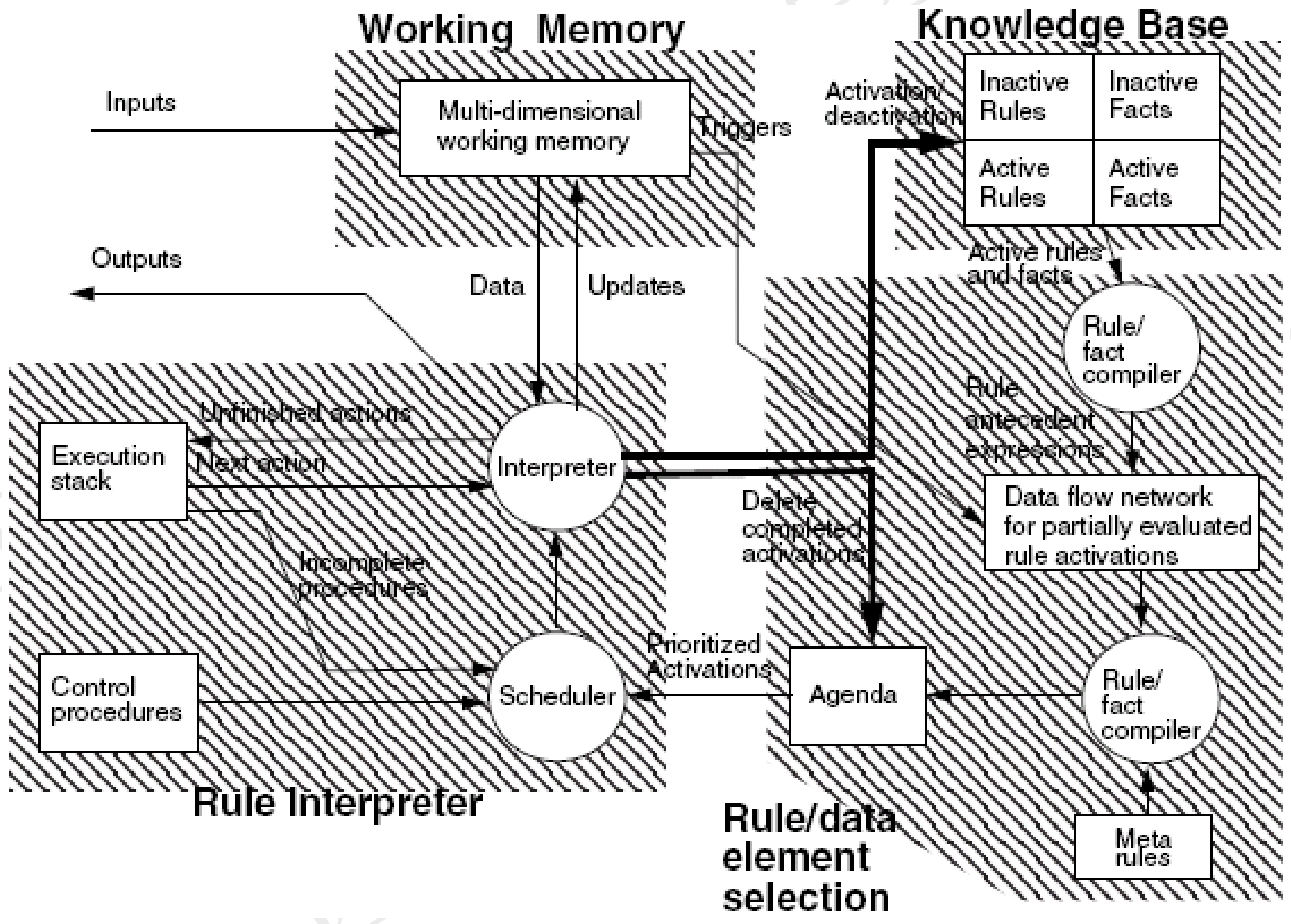
规则系统特征

- Code to be executed (knowledge base)
- Interpretation engine (rule interpreter)
- Control state of interpreter (rule/data selection)
- Current state of the code (working memory)

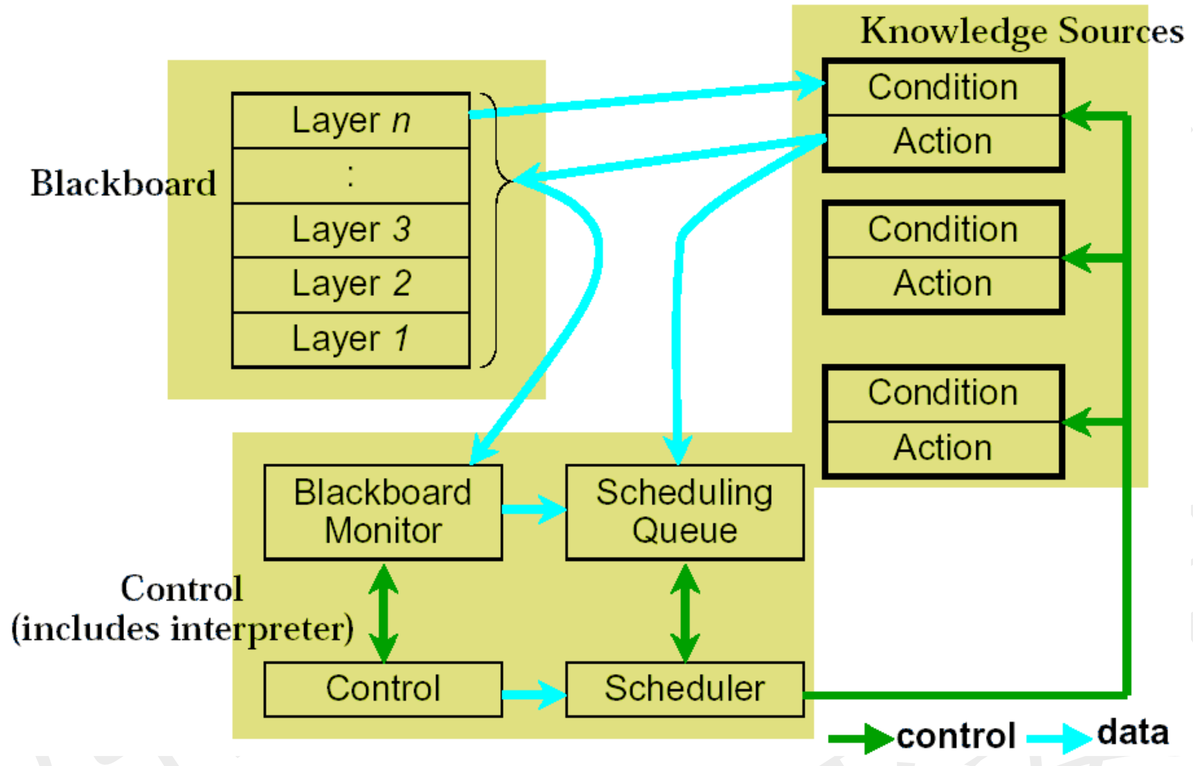
复杂规则系统示例

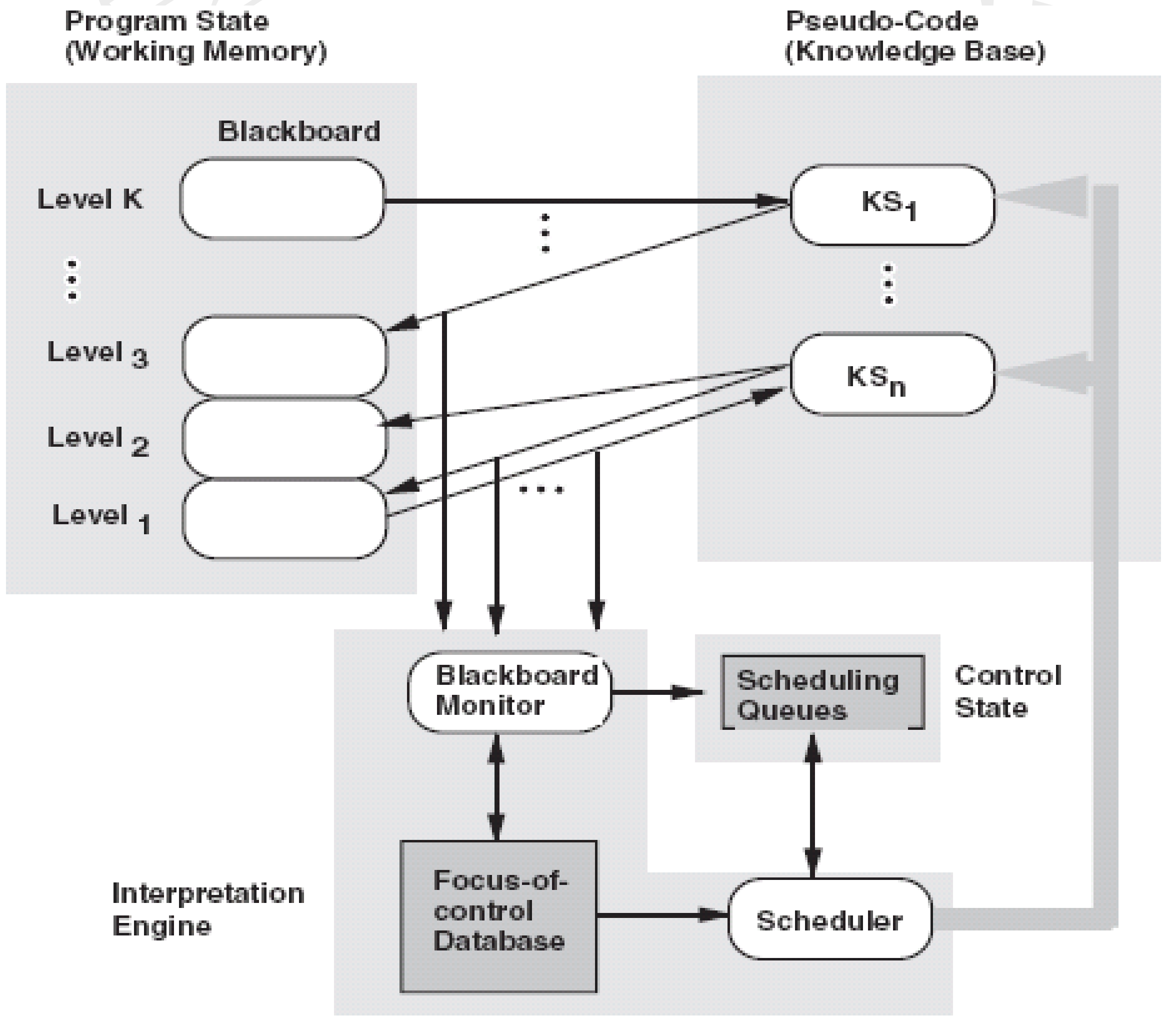

Inputs
Multi-dimensional working memory
Triggers
Activation/ deactivation
Inactive Rules
Inactive Facts
Active Rules
Active Facts
Outputs
Data
Updates
Active rules and facts
Rule/ fact compiler
Rule antecedent expressions
Unfinished actions
Execution stack
Next action
Interpreter
Delete completed activations
Data flow network for partially evaluated rule activations
Incomplete procedures
Prioritized Activations
Control procedures
Scheduler
Agenda
Rule/ fact compiler
Meta rules

规则系统的抽象


Working Memory
Knowledge Base
Inputs
Multi-dimensional working memory
Triggers
Activation/ deactivation
Inactive Rules
Inactive Facts
Active Rules
Active Facts
Outputs
Data
Updates
Active rules and facts
Rule/ fact compiler
Rule antecedent expressions
Unfinished actions
Execution stack
Next action
Interpreter
Data flow network for partially evaluated rule activations
Delete completed activations
Incomplete procedures
Prioritized Activations
Control procedures
Scheduler
Agenda
Rule/ fact compiler
Meta rules
Rule Interpreter
Rule/data element selection

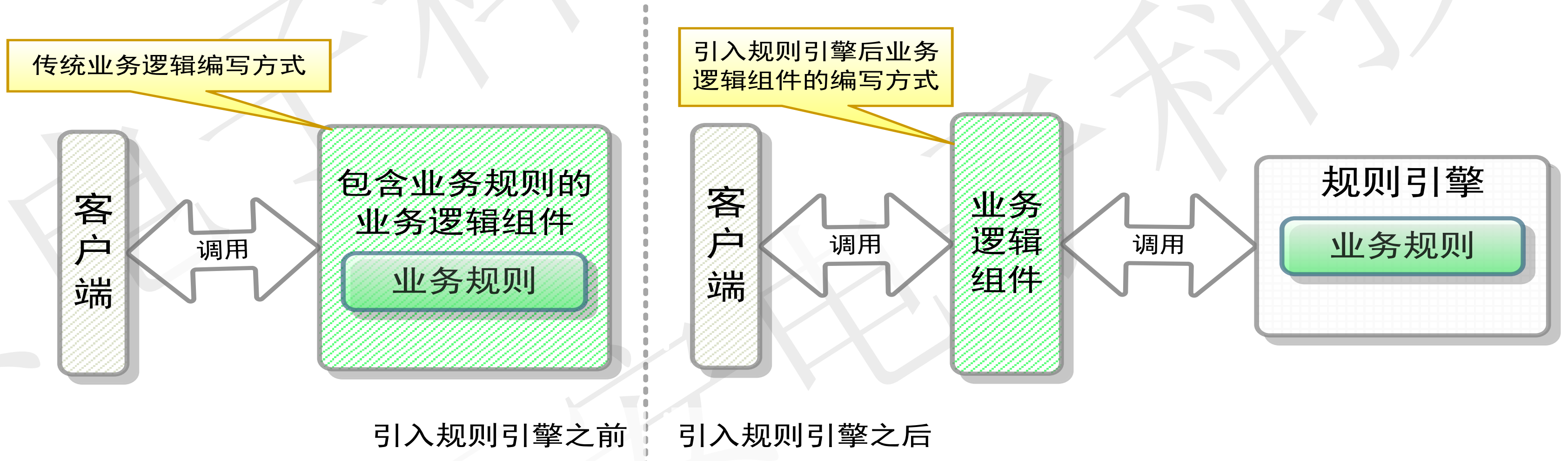
黑板系统与规则系统的相似性





一个实际的规则引擎：Drools

- Drools是Redhat公司JBoss业务逻辑智能模块
- www.jboss.org/drools/
- Drools是处理规则的专家系统



一个实际的规则引擎：Drools

- 声明式编程
  - 规则引擎允许你说“做什么”，而不是“怎样去做”
- 规则系统能解决非常困难的难题
- 逻辑和数据分离
- 快捷和灵活
- 知识集中化
- 工具集成
- 良好的解释机制
- 易于理解的规则

一个例子

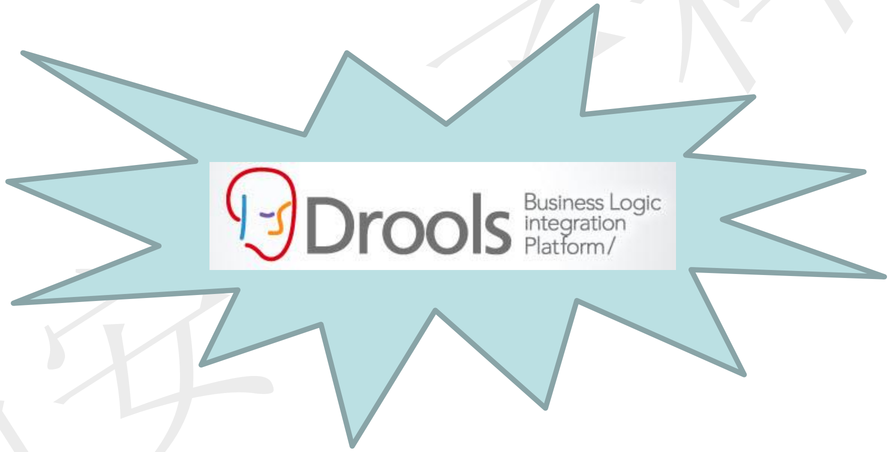
Java语言表示，如果有一个人的名字是“Joe”，而且是个男性，就会输出他的名字跟性别。

```
If ( “Joe”.equal( people.getName() ) )
{
    if( “Male”.equal( people.getSex() ) )
    {
        System.out.priltln(“This is a man, name is Joe.”);
    }
}
```

```
rule "GoodBye"
   when
      People( name = ”Joe”, sex = ”Male”)
   then
      System.out.println((“This is a man, name is Joe.”);
end
```

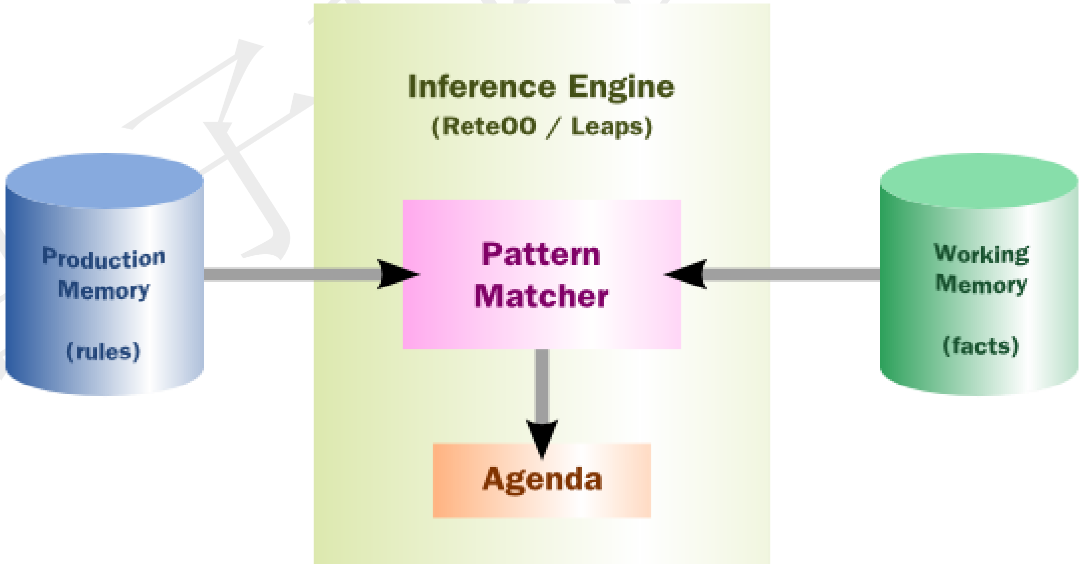
产生背景

- 系统庞大，业务复杂
- 业务规则经常改变
- 24小时服务
- 业务统一管理
- 降低系统维护升级成本

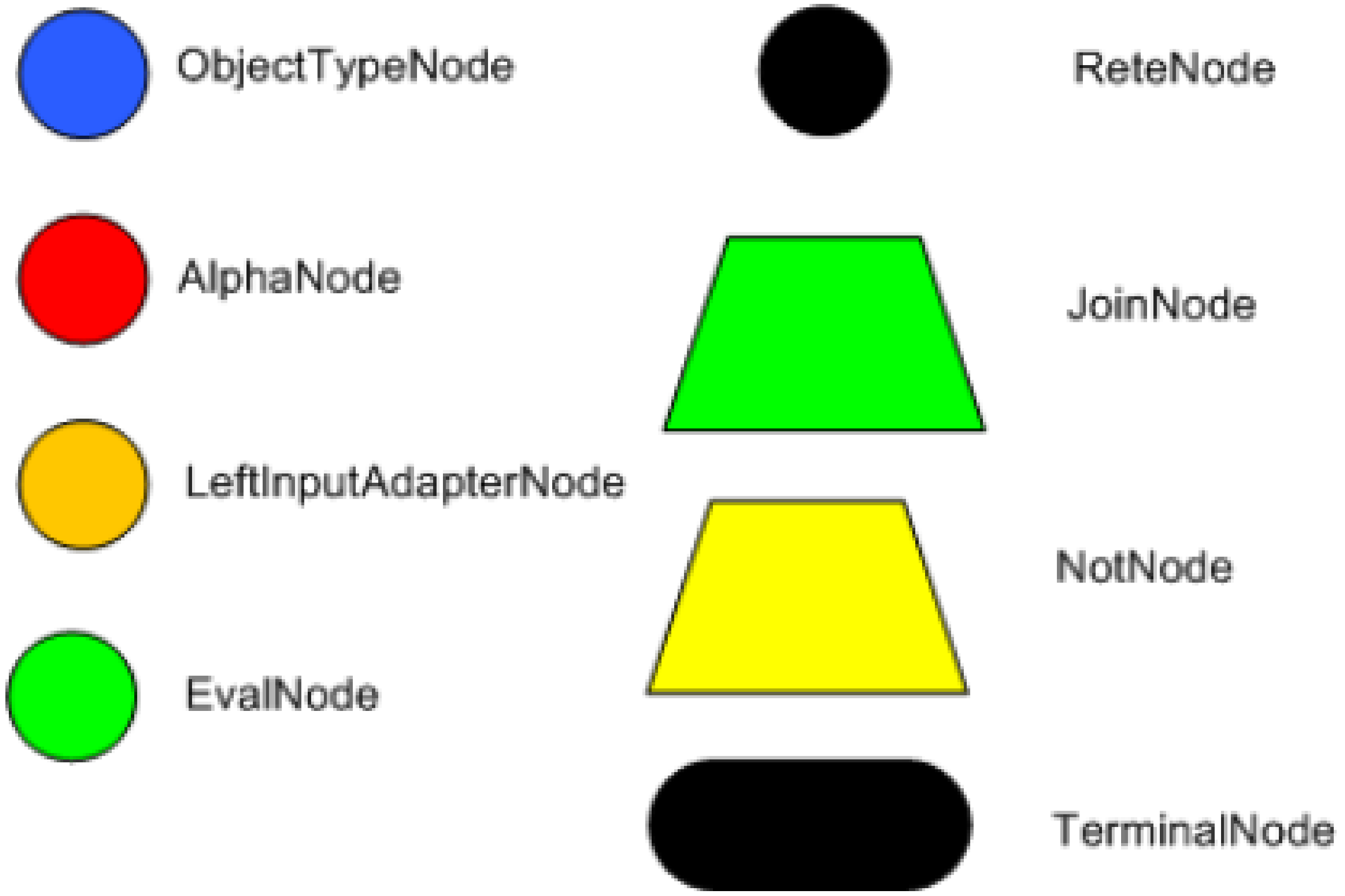


产生式规则引擎

- 许多产品规则系统的大脑实际上就是一个推理引擎，用于匹配事实（Fact）和规则（Rule）
- 当匹配被找到，规则对应的动作（Action）会被触发（Fire）
- 动作
  - 经常会改变事实的状态，或者
  - 在应用上执行一些“外部”动作

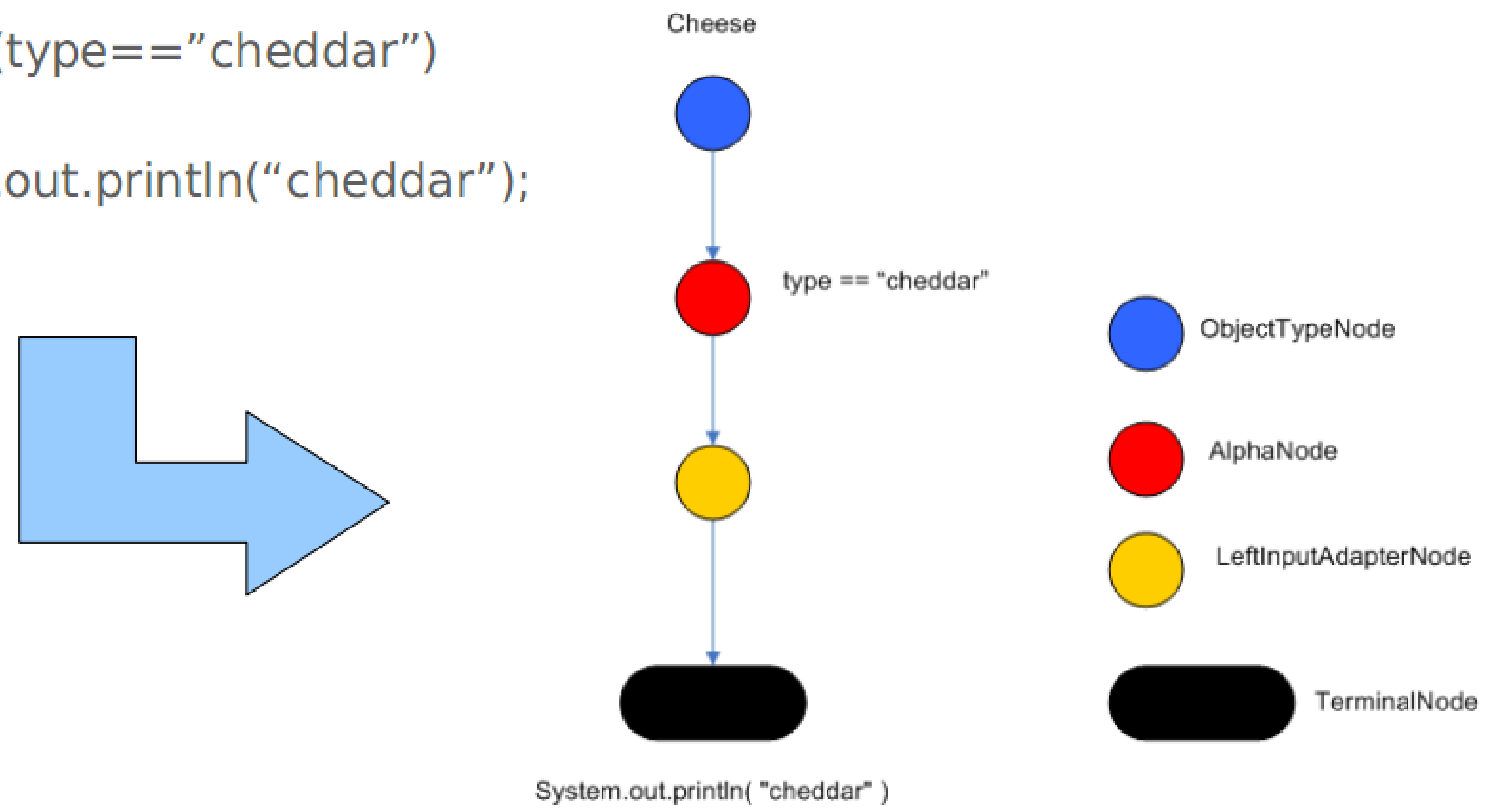


RETE网络节点

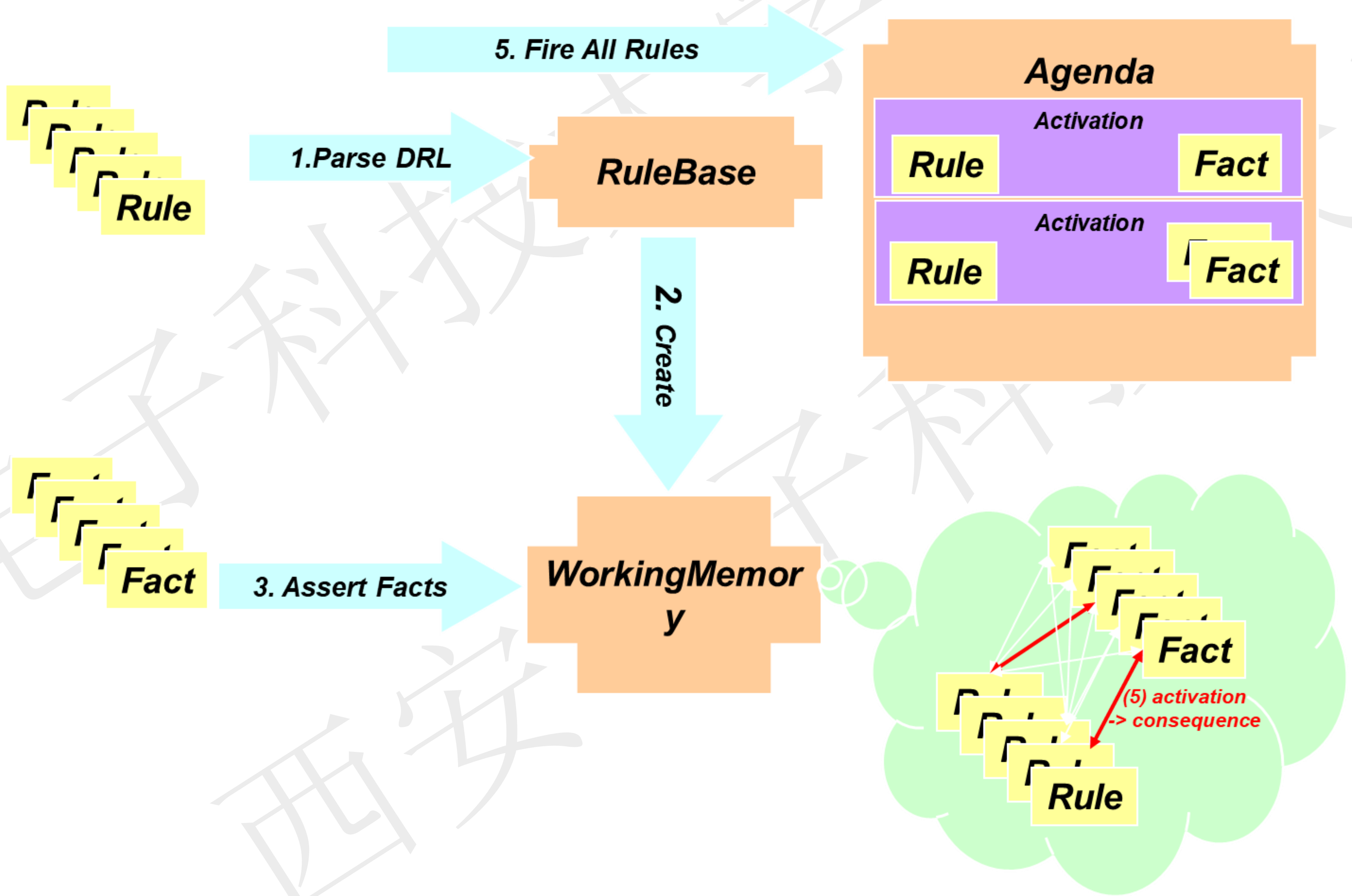

ObjectTypeNode
AlphaNode
LeftInputAdapterNode
EvalNode
ReteNode
JoinNode
NotNode
TerminalNode

简单的RETE网络

```
When
    Cheese(type==”cheddar”)
Then
    System.out.println(“cheddar”);
```

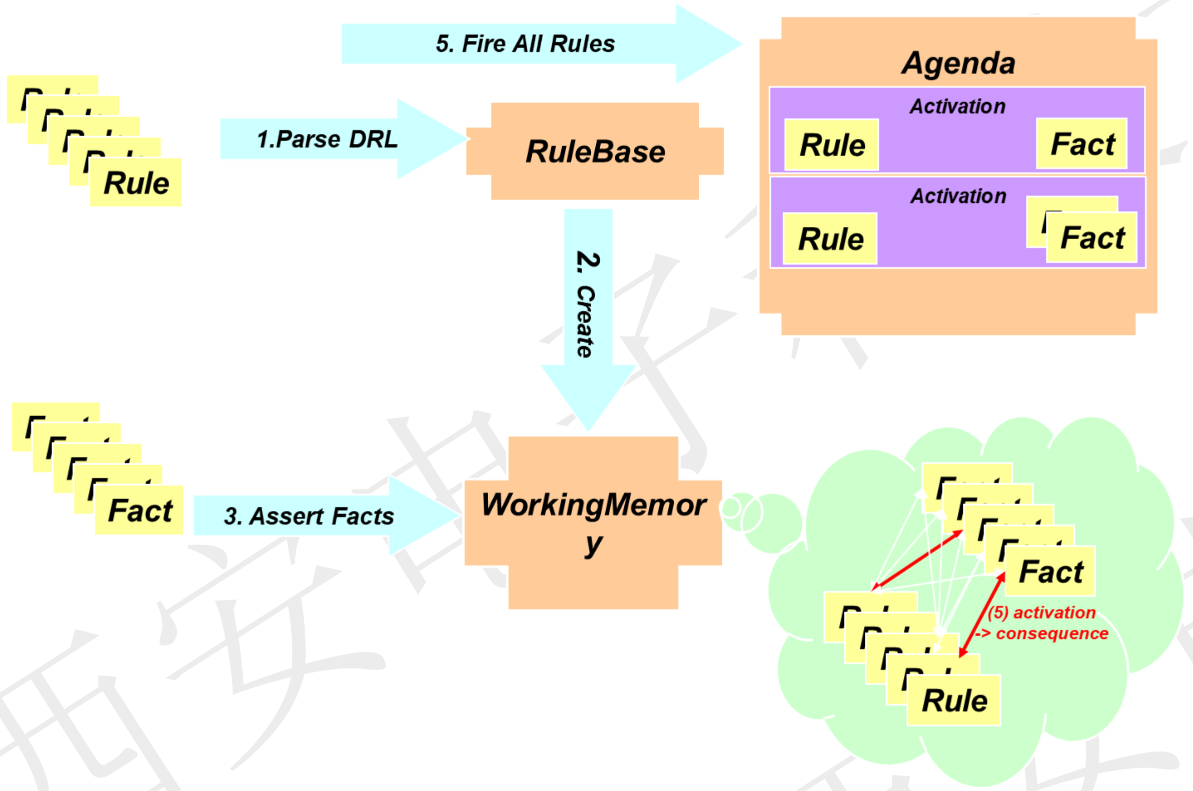


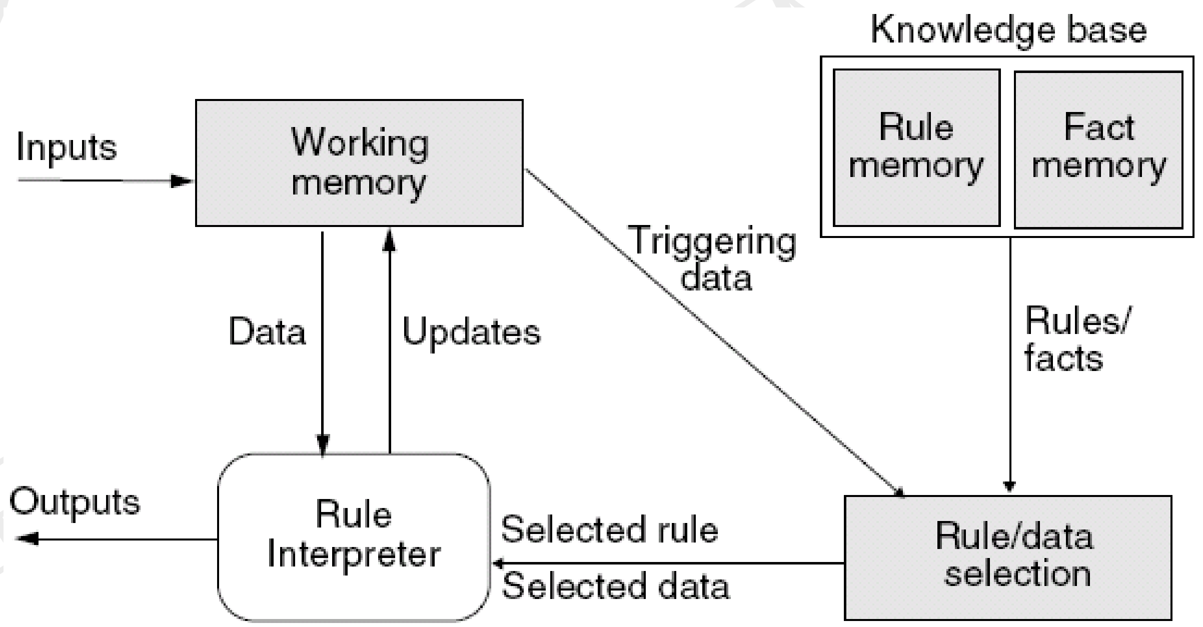
Drools体系结构



5. Fire All Rules
Agenda
Activation
Rule
Fact
Activation
Rule
Fact
Rule
1.Parse DRL
RuleBase
2. Create
Fact
3. Assert Facts
WorkingMemory
Fact
Rule
(5) activation -> consequence

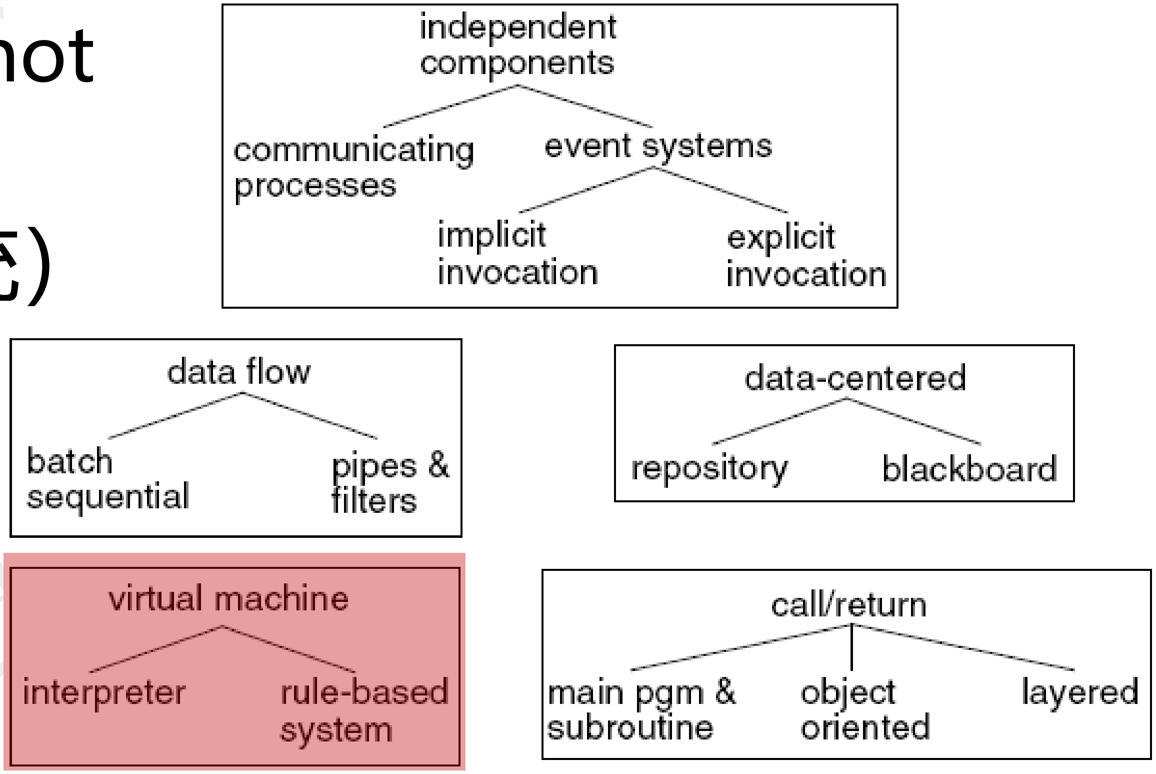
Drools体系结构与经典的规则引擎对比





虚拟机风格总结

- Interpreters（解释器）
  - Simulate functionality which is not native to the hardware
- Rule-based systems（规则系统）
  - Specialization of an interpreter
- Other
  - Syntactic shells
  - Command language processors



西安電子科技大學
XIDIAN UNIVERSITY

谢谢大家！

计算机科学与技术学院微信

课程讨论群

鲍亮 副教授 博导

邮箱：baoliang@xidian.edu.cn

主页：https://web.xidian.edu.cn/yslin/