Script Development

A script, in OTAWA, allows to configure a WCET computation. It contains :

1.step-by-step code processor invocations（调用）,

2.a way to let the user fine-tune（微调） the computation thanks to user parameters,

3.description of the involved architecture,

4.possible documentation about the computation (allowing to inform the user about limitations for example).

The scripts are described in [XML format](http://www.w3.org/XML/) and, therefore, easy to read and write by a human user. They are mostly the simpler way to extend OTAWA without the need to understand the internal API of the framework. In addition, scripts are easy to write because they use well-known formats based on XML like [XInclude](http://www.w3.org/TR/2006/REC-xinclude-20061115/) or [XSLT](http://www.w3.org/TR/xslt20/) that gives a lot of power in the script behavior.

 two usage：

（1） used to configure the computation for a particular architecture. Instead of using the main stream computation approach, they allows to easily perform and automate specific analyzes to fine-tune a computation.

（2）to provide support in OTAWA for a new micro-architecture: indeed, they allows to describe the components of an architecture or a model processor (pipeline, caches, memory space) and then to invoke all analyzes required to support the architecture.

1.1Notation

The description of the XML files in this document merges [XML](http://www.w3.org/XML/) textual format with [EBNF](http://fr.wikipedia.org/wiki/Extended_Backus-Naur_Form)（扩展的巴科斯范式）.

The grammar is formed of a list of rules whose root is the first one. Each rule is made of:

an XML comment giving the name of the rule,

the matching XML element possibly containing other elements giving the shape of the rule .

1.2 Script Format

A script is a textual XML file whose extension is usually .osx for Otawa Script XML. It follows the usual rule of XML and the top-level element is called otawa-script:

<!-- OTAWA-SCRIPT ::= -->

<?xml version="1.0"?>

<otawa-script

xmlns:xi="http://www.w3.org/2001/XInclude"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<!-- DESCRIPTION -->

<!-- IDENTIFICATION -->

<!-- CONFIGURATION -->?

<!-- PLATFORM -->?

<!-- SCRIPT -->?

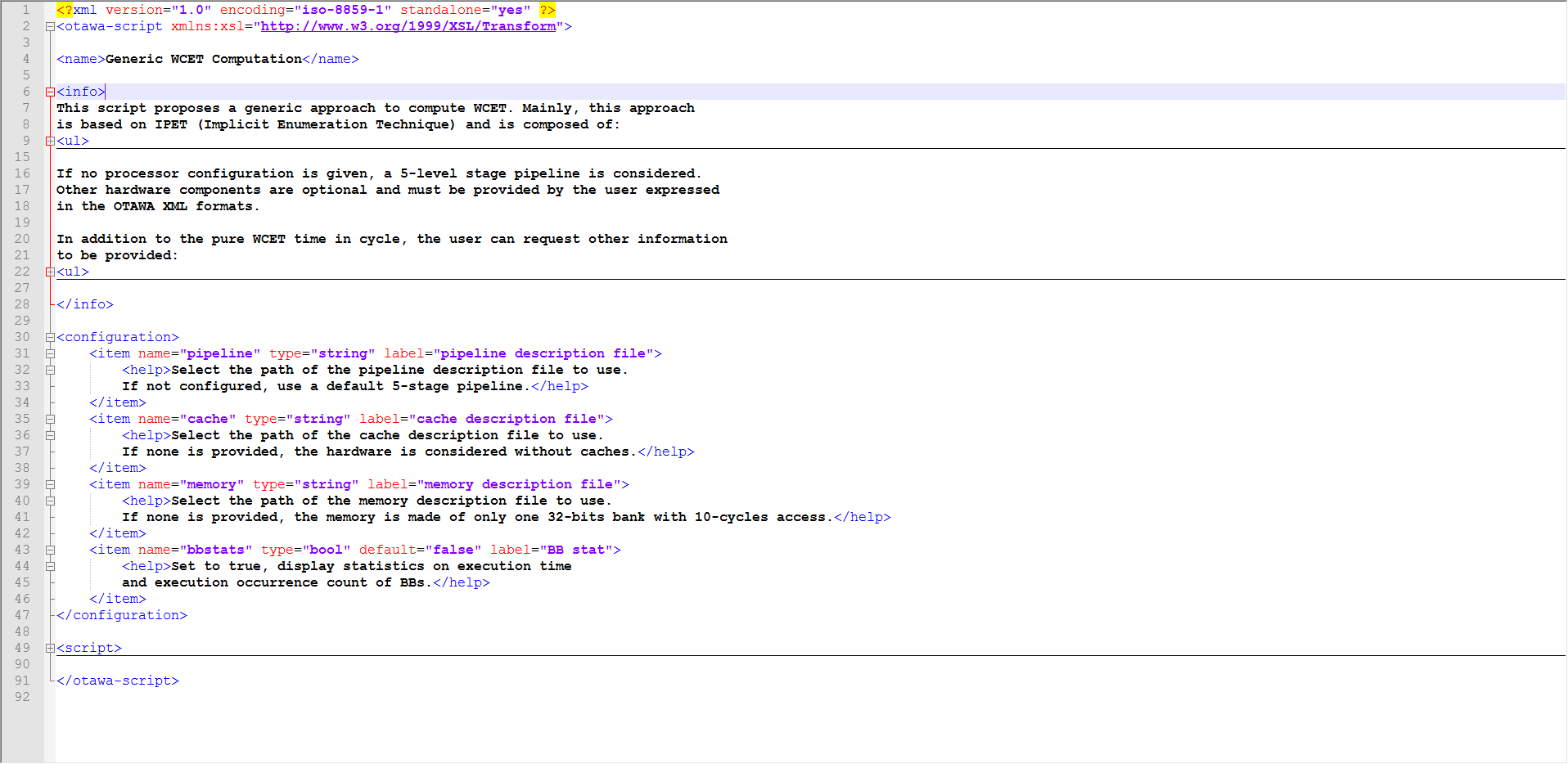
DESCRIPTION  provides various information targeting the human user,

IDENTIFICATION contains mainly identifier about the hardware (architecture, model, ABI),

CONFIGURATION provides a list of items the user may tune,

PLATFORM describes the hardware,

SCRIPT details the performed computation steps.

1.2.1Description

The description is made of the following items:

<!-- DESCRIPTION ::= -->

<name> TEXT </name>

<info> XHTML </info>?

<path to="PATH"/>\*

The name tag is mandatory and provide the name of the script as displayed to the human user.

The info element may contain a whole documentation describing the script, its applications and its limitation. As it is intended displayed to the human user and as it may contains a structured documentation, it is described in [XHTML](http://www.w3.org/TR/xhtml1/).

Finally, the path is used for the internal work of the script inside OTAWA.

1.2.2Identification

The identification part has the following structure:

<!-- IDENTIFICATION ::= -->

<id>

<arch>TEXT</arch>

<abi>TEXT<abi>

<mach>TEXT</mach>

</id>

The arch tag allows to identify the progamming model, also called the ISA (Instruction Set Architecture) of the supported hardware. Common values include arm, powerpc, sparc, x86, etc. （指令架构）

The abi element gives the Application Binary Interface with common values being eabi, elf, linux, etc. （应用程序二进制接口）

Finaly, the mach element allows to precisely identify the processor model the script is targetting. （针对的处理器模型）

Only the arch element is mandatory to be able to check if it supports the instruction set used in the processed executable file.

1.2.3 Configuration Description

The configuration lists a set of items to let the human user parameterize the computation:

<!-- CONFIGURATION ::= -->

<configuration>

<!-- CONFIGURATION-ITEM -->\*

</configuration>

<!-- CONFIGURATION-ITEM ::= -->

<item

name="TEXT"

type="bool|int|string|range|enum"

default="TEXT"?

label="TEXT"?>

<help> <!-- TEXT --> </help>

</item>

--an internal name used to identify the variable containing their value in XSLT,

--a label, the name of the configuration item displayed to the user,a default value,

--a type that describes the type of value,

--an help sub-element that contains human-readable information to help the user understanding the configuration item.

In addition, each type of items may have its own set for attributes and sub-elements.

1.2.4 Platform Description

The platform provide details about the hardware feature of the targeted system.

<!-- PLATFORM :: -->

<platform>

<!-- PIPELINE -->?

<!-- CACHES -->?

<!-- MEMORY -->?

</platform>

The items found in the PLATFORM may described directly in the script or in a separate file included by XInclude.

In the latter case, the file must be prefixed by the usual XML identification line:

<?xml version="1.0"?>

The example belows uses XInclude to get the hardware description from three different external files:

<platform>

<xi:include href="mpc5554/pipeline.xml"/>

<xi:include href="mpc5554/cache.xml"/>

<xi:include href="mpc5554/memory.xml"/>

</platform>

Notice that the relative paths passed in the href attribute are resolved from the XML base of the document, that is, the directory containing the script file.

1.2.5 Script Description

This element describes the script itself, that is, the analyzes to apply to get the WCET. In fact, the script works directly on the code processor structure of OTAWA that is composed:（应用于获取WCET的分析，直接在OTAWA的代码处理器结构上工作）

--code processors that implements simple analyzes or may transform the

program representation（实现简单的分析或者可以转换程序表示）

--features are required or provided by code processors and represent information retrieved by the analyzes（代码处理器提供，代表由分析检索得到的信息）

--properties that are annotations representing results of analyzes, hooked to the

program representation and grouped in features.（属性是表示分析结果的注解，挂钩到程序表示，分组在features中）

Code processors, features and properties are documented in the automatic documentation of OTAWA.

In OTAWA, computing the WCET is invoking either the code processor computing the WCET, or requiring the feature provided by this code processor.（调用计算WCET的代码处理器，或要求该代码处理器提供feature） In turn, this processor may require other features that will be achieved by other code processors and so on. The rule is that if an already-provided feature is required, it is used as is. If it is not provided, the default processor associated to the feature is invoked.

It comes out that the order of feature requirements or code processor invocations matters! To substitute an analysis A to the default analysis B of a feature F, the A analysis must be invoked first to let it providing the feature F. When a code processor will require the feature F, it will use the feature provided by A as it is already available.（出现 feature请求或代码处理器调用的顺序很重要。对于一个需要得到的feature F，要将analysis A替换为默认的analysis B，必须首先调用analysis A以提供feature F。当代码处理器需要feature F时，它将使用A提供的feature）

The script part has the syntax below and is made of a sequence of steps, possibly with configuration items that apply to all steps:

<!-- SCRIPT ::= -->

<script>

<!-- CONFIG -->\*

<!-- STEP -->+

</script>

A step may invoke a code processor (attribute processor) or require a feature (attribute feature). If a step contains configuration items, they are only applied to this step and to code processors automatically invoked from this step.

<!-- STEP ::= -->

<step processor="C++PATH"? require="C++PATH"?>

<!-- CONFIG -->\*

</step>

The C++PATH used to identify a processor or a feature (but also a property) is the full-qualified path of the object in the C++ implementation of OTAWA. For example, if a code processor implementing class is MyAnalysis that is contained in namespace my and otawa, the matching C++PATH is otawa::my::MyAnalysis.

Example：LPC2138 microprocessor

<script>

<step require="otawa::VIRTUALIZED\_CFG\_FEATURE"/>

<step processor="otawa::lpc2138::CATMAMBuilder"/>

<step processor="otawa::lpc2138::ARM7ParamExeGraphBBTime">

<config name="otawa::lpc2138::FLASH\_MISS" value="56"/>

</step>

<step require="otawa::ipet::WCET\_FEATURE"/>

</script>

This script requires first the feature otawa::VIRTUALIZED\_CFG\_FEATURE ensures that all functions calls have been inlined. In fact, this requirement will cause the invocation of several analyses like the flow fact loader, program text decoder, CFG building, etc.

Then the otawa::lpc2138::CATMAMBuilder analyzes the prefetcher of the LPC2138 flash memory and computes the execution of the blocks with otawa::lpc2138::ARM7ParamExeGraphBBTime. In this step, a configuration parameter is passed to configure the time for a memory flash miss. As will be presented below, the value is rarely a constant: it may be derived from the configuration variables.

Finally, the WCET computation is required, otawa::ipet::WCET\_FEATURE, that will build the ILP system, flowfact constraints but will re-use the block timings already provided by otawa::lpc2138::ARM7ParamExeGraphBBTime without invoking the default computation of block timings.