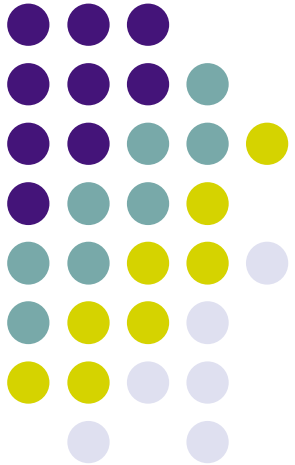
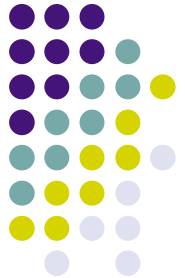
Verifying CIM Models of Apache Web-Server Configurations

Carsten Sinz **Amir Khosravizadeh** **Wolfgang Küchlin**
Symbolic Computation Group, WSI for Computer Science
University of Tübingen, Germany

Viktor Mihajlovski
Linux Technology Center (LTC)
IBM Laboratory Böblingen, Germany

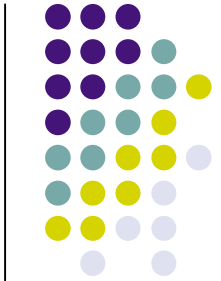


Outline



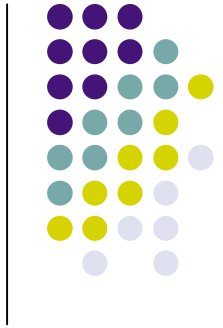
- Motivation
- Apache Web-Server Configuration
- Common Information Model (CIM)
- CIM Constraint Language $\mathcal{CCL}$
- Verification Framework

Motivation (I)



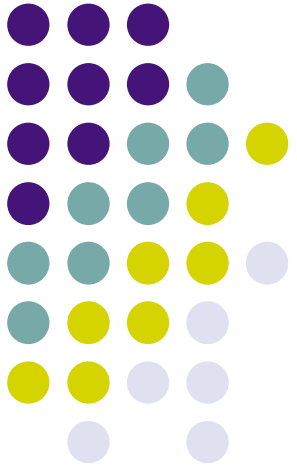
- More than 14 million installations of Apache Web-Server (according to Netcraft survey)
- Web-server configuration is a complex and demanding task:
  - *Security issues*: access rights/permissions; logging
  - *Performance issues*: process/thread management
  - *Site specific local restrictions*: user policies, handling of CGI/SSI
- Goal: Automatically checking the “correctness” (“consistency”) of a configuration

Motivation (II)

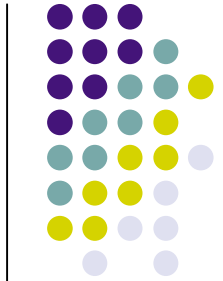


- Correctness issues:
  - *Security/ Privacy*: No possibility to by-pass access restrictions; access only for authorized users (several virtual hosts on one machine)
  - *Stability*: No server crashes by unintended mal-configuration
- Verification Environment:
  - Allows checking of a given configuration for global & site specific constraints
  - Suggests actions to repair flawed installation
  - Provides constraint editor to ascertain non-contradictory constraints

Apache Web-Server Configuration

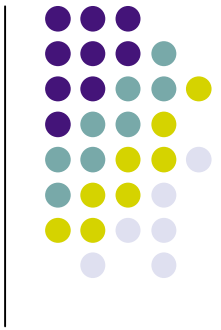


Apache Configuration



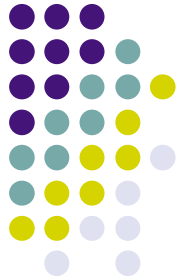
- Different configuration files:
  1. **httpd.conf**: server's main configuration file
  2. **.htaccess**: local access rights (per directory)
  3. **srm.conf / access.conf** [up to Version 1.3]: namespace and resources (file typing, aliases) / global access rights
- Apache nomenclature:
  - *Directive*: (textual representation of a) configuration option
  - *DocumentRoot*: place where Web pages and other content is stored)
  - *VirtualHost*: one of several logical Web-servers managed by A.
- More than 200 different (configuration) directives

Configuration File Example (I)



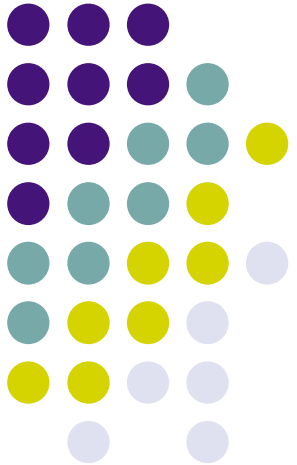
```
ServerRoot "/apache"              # root dir. containing config./log files,...
Timeout 300                       # number of seconds before timeout is sent
KeepAlive On                      # persistent connections?
StartServers          5           # number of servers at initialization
MinSpareServers       5           # minimal/maximal number of servers that
MaxSpareServers      10           #   are kept spare
Listen 80                         # port number for incoming requests
LoadModule mod_access.so          # dynamically loaded modules
...
LoadModule mod_rewrite.so
User nobody                       # user/group name under which the server
Group nobody                      # is running
ServerAdmin  admin@host
ServerName   www.company.com
DocumentRoot "/apache/htdocs" # base of all web contents
```

Configuration File Example (II)

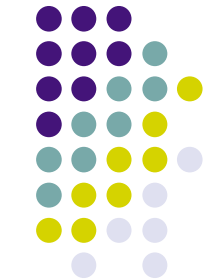


```
<Directory />                          # access rights for FS root directory
    Options FollowSymLinks
    AllowOverride None                 # no changes allowed in .htaccess
</Directory>

<Directory "/apache/htdocs">           # access rights for DocumentRoot
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

UserDir public_html                    # users' home directories

DirectoryIndex index.html index.html.var
AccessFileName .htaccess

<Files ~ "^\.ht">                      # .ht* files are not viewable
    Order allow,deny
    Deny from all
</Files>
```

Common Information Model (CIM)

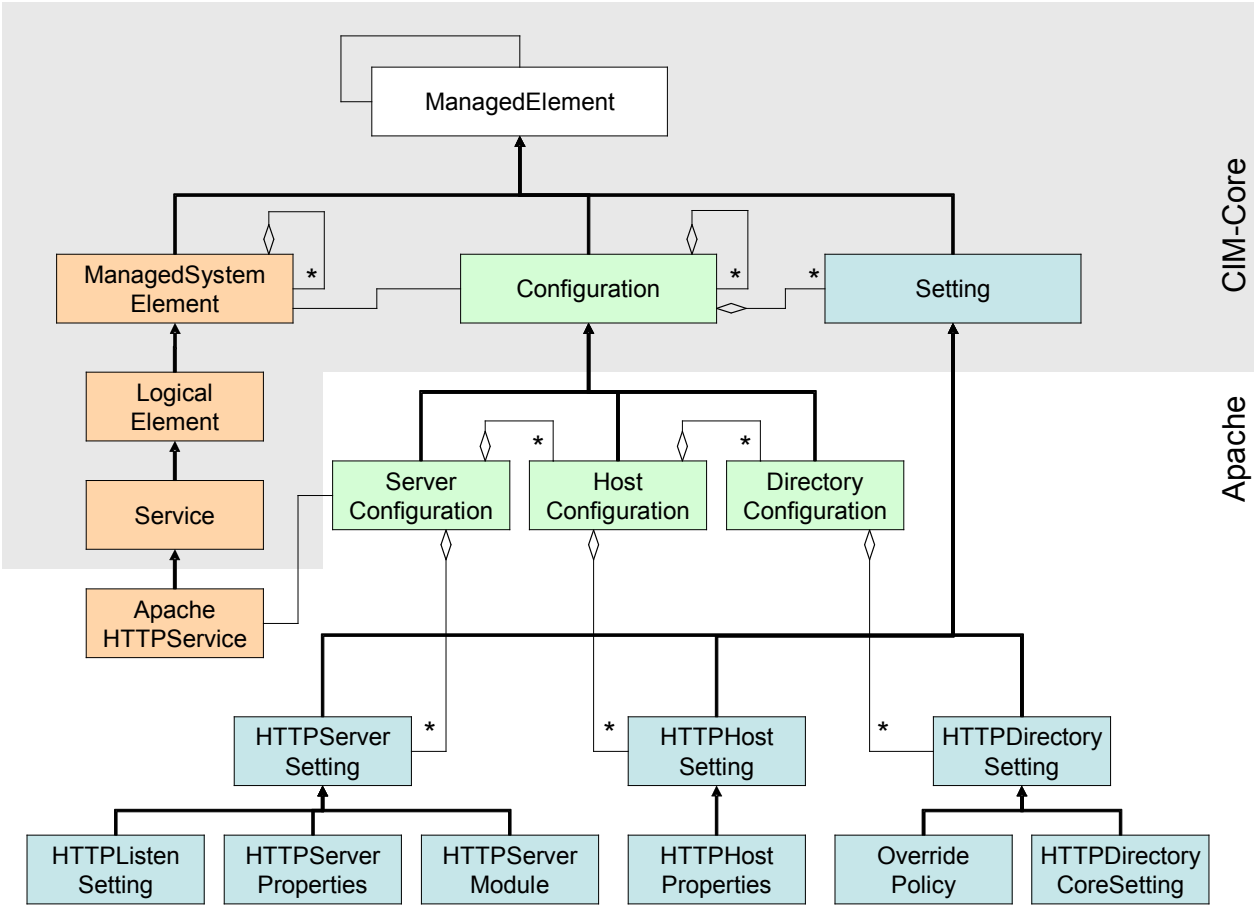


Common Information Model (CIM) Basics

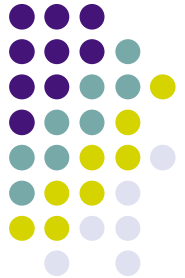


- CIM: Object-oriented data model (classes/instances) for system management purposes
- Aims at replacing several special-purpose protocols:
  - DMI (desktop management interface)
  - SNMP (simple network management protocol)
  - CMIP (common management information protocol)
- Graphical representation: UML diagrams
- Textual representation: MOF (managed object format, similar to CORBA IDL)
- CIM *Schema*: Collection of CIM classes

CIM Classes for Apache Configuration

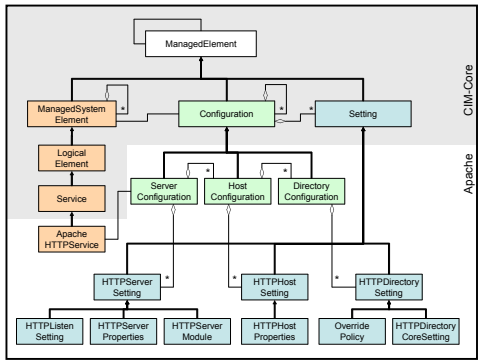


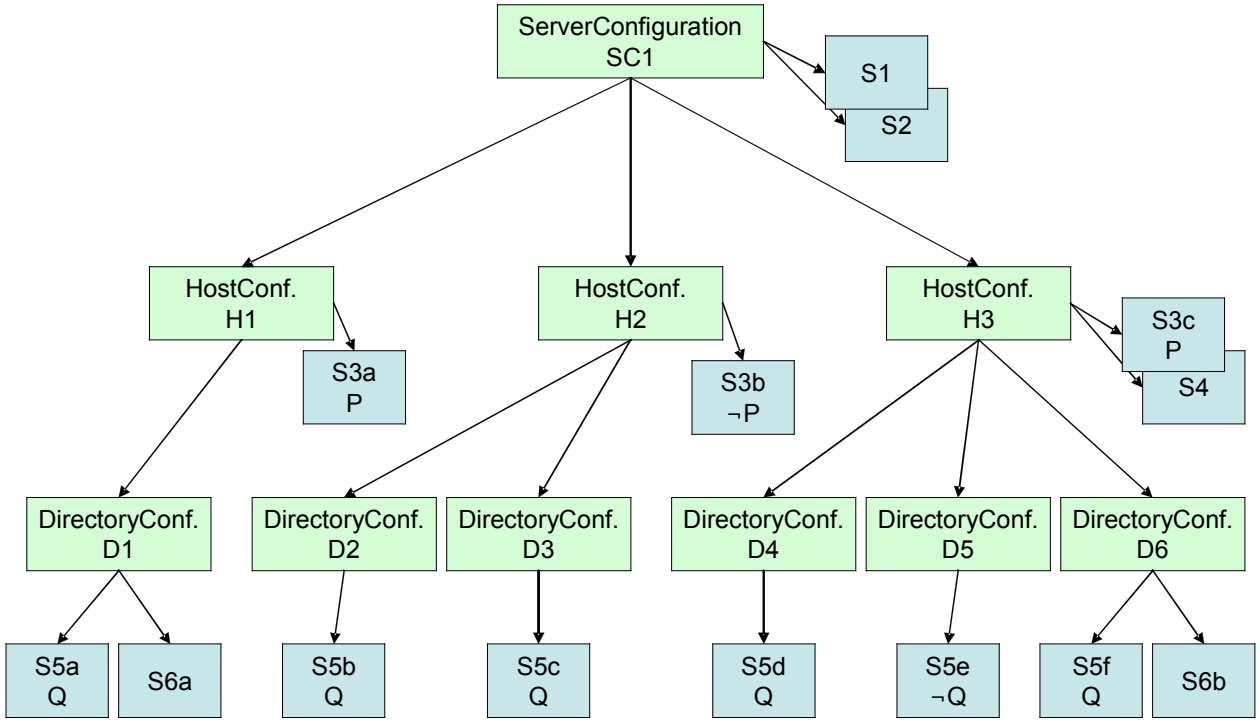
MOF Representation of Apache CIM Classes



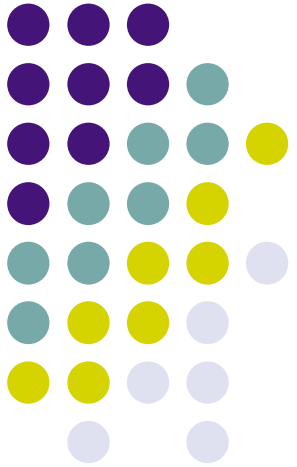
```
class Apache_HttpHostConfiguration : CIM_Configuration
{
   [Key] String Name;
};
class Apache_HttpServerProperties : Apache_HttpServerSetting
{
   [Key] String ConfigName;
   String BindAddress;
   String CoreDumpDirectory;
   uint16 MaxClients;
   uint32 MaxRequestsPerChild;
   int16 MaxSpareServers;
   int16 MinSpareServers;
   ...
};
```

Hierarchy of Configuration and Setting Instances

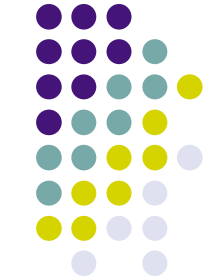




CIM Constraint Language CCL



CCL: Syntax



v-expressions (denoted by *s*,*t*,…):

C.P	where C is a class name and P a property name.
C.$\langle P_1,\ldots,P_k \rangle$	where C is a class name and $P_1,\ldots,P_k$ are property names.
v	where v is an arbitrary property value constant (string, number..)
f($s_1,\ldots,s_k$)	where f is a k-ary (interpr.) function and $s_1,\ldots,s_k$ are v-exprs.

a-expressions:

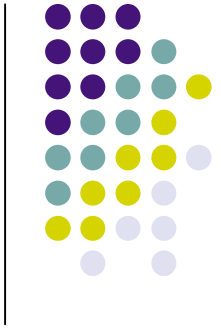
R($s_1,\ldots,s_k$)	where R is a k-ary (interpr.) predicate and $s_1,\ldots,s_k$ are v-exprs.
$\exists^{\geq n}$C	where *n* is a natural number and C a class name.
$\exists^{\geq n}$C.P	where *n* is a natural number, C a class name and P a property.

f-expressions (denoted by *F*,*G*,…):

Boolean logic expressions built from connectives $\wedge$, $\vee$, $\neg$, true, false, $\leftrightarrow$, $\rightarrow$ and auxiliary symbols (and), using a-expressions as atoms.

[C]*F*	where C is a class name and *F* an f-expression.

CCL: Semantics



v-expressions [values]:

C.P	all values occurring under property P of any instance of class C
$C.\langle P_1,\ldots,P_k\rangle$	all tuples $(v_1,\ldots,v_k)$ occurring under properties $(P_1,\ldots,P_k)$
v	value constant
$f(s_1,\ldots,s_k)$	appl. of f to the set of values $s_1,\ldots,s_k$ (yielding a set of values)

a-expressions [atoms]:

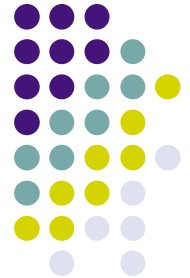
$R(s_1,\ldots,s_k)$	is true, if $s_1,\ldots,s_k$ are in relation R
$\exists^{\geq n}C$	is true, if there are at least *n* instances of class C
$\exists^{\geq n}C.P$	is true, if there are at least *n* instances of class C with property P defined

f-expressions [formulae]:

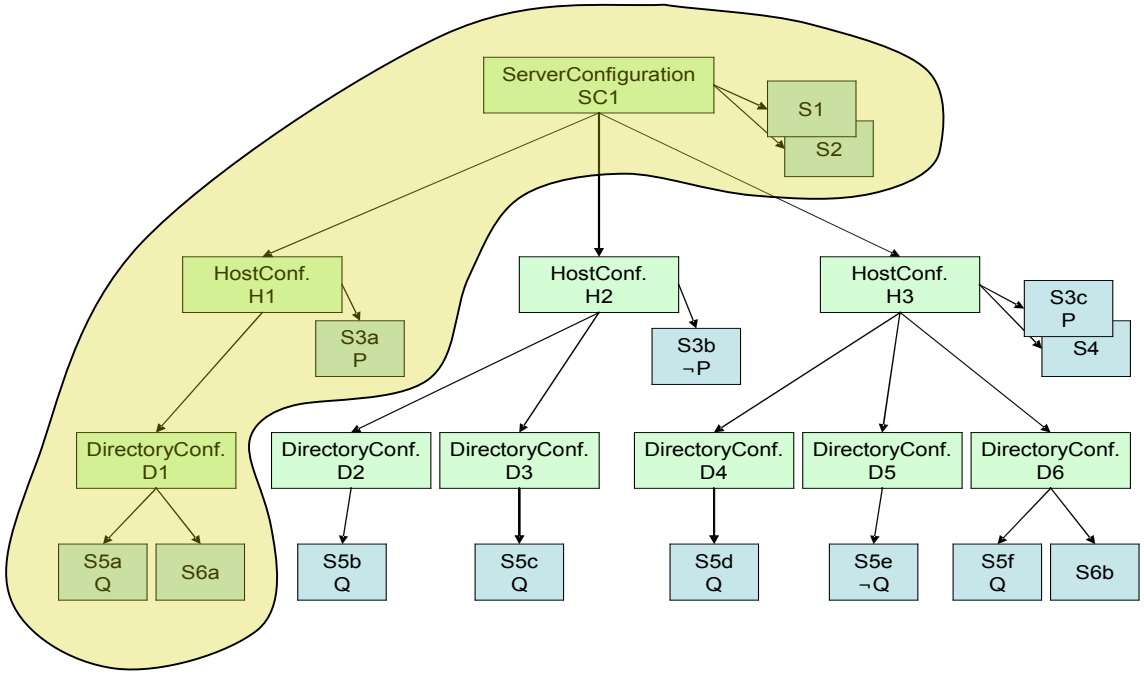
[C]*F*	is true, if *F* holds in context C ("universal path quantification")

Formal semantics in paper.

CCL: Context operator (Example)

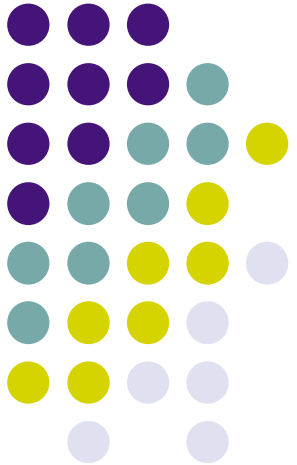


Relevant settings (the *context*) for directory D1?

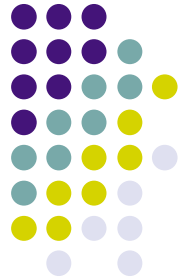


[C]*F*: Filters set of considered class instances

Verification of Configurations



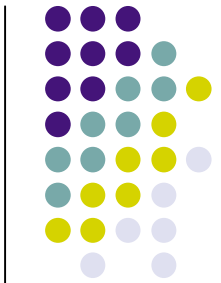
Sample Apache Configuration Constraints



- The *ServerRoot* directive must be specified exactly once in the server configuration.
- The following restrictions have to hold:

  MaxSpareServers > *MinSpareServers* and *MinSpareServers* > 1.
- When several virtual hosts are running on the same server, each of them must have its own unique *ServerName*.
- For security and privacy reasons it is strongly recommended that the log files of all virtual hosts are not visible to the outside world. For example, the *ErrorLog* file should not be located in *DocumentRoot* or a subdirectory thereof.
- All virtual servers should have their own log files.

Configuration Constraints in *CCL* (I)



1. **$\exists^{=1}$ServerProperties.ServerRoot**

   Property ServerRoot is defined exactly once.

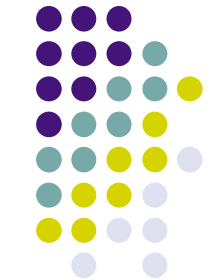
2. **[ServerConfiguration](ServerProperties.MinSpareServer < ServerProperties.MaxSpareServer) ∧ [ServerConfiguration](ServerProperties.MaxSpareServer > 1)**

   For each server configuration, *MinSpareServer* and *MaxSpareServer* properties are set correctly. The comparison operators < and > are relation symbols contained in set R.

3. **|HostProperties.ServerName|=|HostConfiguration.Name|**

   Each virtual host has its own unique server name. We use an additional unary function, |·|, computing the cardinality of its argument set.

Configuration Constraints in *CCL* (II)



4. **[HostProperties]¬isPrefixOf(HostProperties.DocumentRoot, HostProperties.ErrorLog)**

   The error log should not be stored in directory DocumentRoot or a subdirectory thereof.

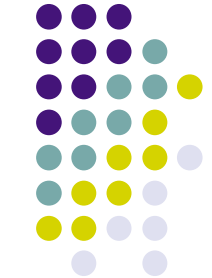
5. **[HostConfiguration] HostProperties.⟨HostAddress,HostPort⟩ ⊆ ListenSetting.⟨ListenAddress, ListenPort⟩**

   The address/port pair of each virtual host must be an address/port that the Web-server is listening to.

6. **∃ServerProperties.ConfigName ∧ ∃ServerProperties.PidFile**

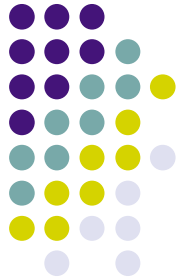
   A configuration name and PID file must be specified for the Web-server.

Verification Environment



- CIM classes for Apache configuration provided by IBM
- Prototypical constraint Checker *CIMVerifier* implemented in Java
  - Uses variant of CCL, called ConQuery
  - Apache configuration data provided by WBEM infrastructure
  - *CIMVerifier* evaluates given set of constraints (as in examples above)
  - Relations and function symbols of CCL are mapped to Java methods and are resolved by Java *Reflection*
  - Violated constraints are reported
  - Most likely mis-configured directives are determined by heuristic
  - Only constraint evaluation, no theorem proving so far

Conclusion



- Verification of complex software has to extend to the configuration phase. (Only correct configuration can ensure correct operation).
- CIM provides convenient, industrially accepted semi-formal intermediate model.
- Constraint evaluation available, theorem proving in $\mathcal{CCL}$ (e.g., for automatic error correction) yet to realize.

Further information:

http://www-sr.informatik.uni-tuebingen.de