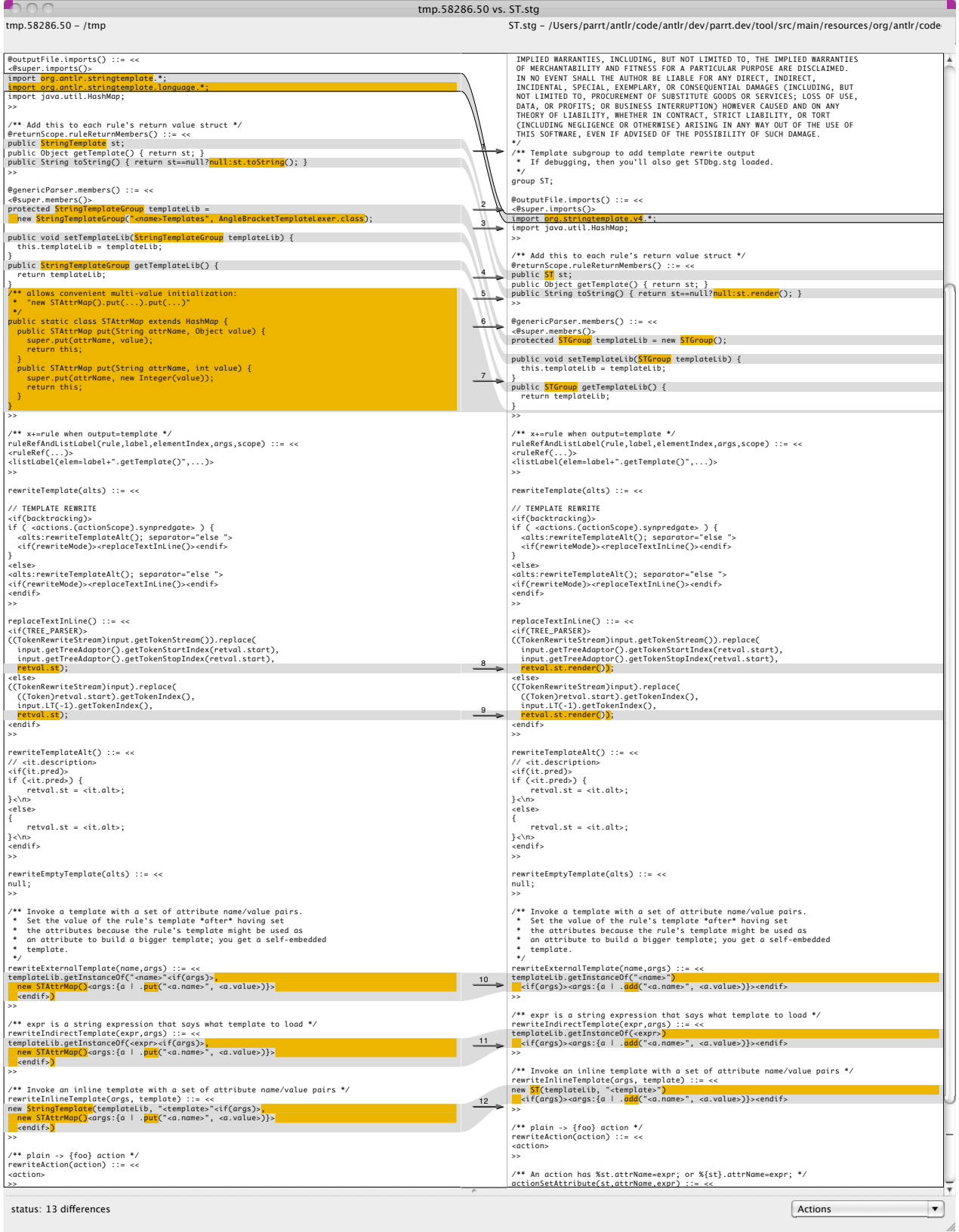
tmp.58286.50 vs. ST.stg
tmp.58286.50 – /tmp
ST.stg – /Users/parrt/antlr/code/antlr/dev/parrt.dev/tool/src/main/resources/org/antlr/code
@outputFile.imports() ::= <<
<@super.imports()>
import org.antlr.stringtemplate.*;
import org.antlr.stringtemplate.language.*;
import java.util.HashMap;
>>
/** Add this to each rule's return value struct */
@returnScope.ruleReturnMembers() ::= <<
public StringTemplate st;
public Object getTemplate() { return st; }
public String toString() { return st==null?null:st.toString(); }
>>
@genericParser.members() ::= <<
<@super.members()>
protected StringTemplateGroup templateLib =
new StringTemplateGroup("<name>Templates", AngleBracketTemplateLexer.class);
public void setTemplateLib(StringTemplateGroup templateLib) {
this.templateLib = templateLib;
}
public StringTemplateGroup getTemplateLib() {
return templateLib;
}
/** allows convenient multi-value initialization:
* "new STAttrMap().put(...).put(...)"
*/
public static class STAttrMap extends HashMap {
public STAttrMap put(String attrName, Object value) {
super.put(attrName, value);
return this;
}
public STAttrMap put(String attrName, int value) {
super.put(attrName, new Integer(value));
return this;
}
}
>>
/** x+=rule when output=template */
ruleRefAndListLabel(rule,label,elementIndex,args,scope) ::= <<
<ruleRef(...)>
<listLabel(elem=label+".getTemplate()",...)>
>>
rewriteTemplate(alts) ::= <<
// TEMPLATE REWRITE
<if(backtracking)>
if (<actions.(actionScope).synpredgate>) {
<alts:rewriteTemplateAlt(); separator="else ">
<if(rewriteMode)><replaceTextInLine()><endif>
}
<else>
<alts:rewriteTemplateAlt(); separator="else ">
<if(rewriteMode)><replaceTextInLine()><endif>
<endif>
>>
replaceTextInLine() ::= <<
<if(TREE_PARSER)>
((TokenRewriteStream)input.getTokenStream()).replace(
input.getTreeAdaptor().getTokenStartIndex(retval.start),
input.getTreeAdaptor().getTokenStopIndex(retval.start),
retval.st);
<else>
((TokenRewriteStream)input).replace(
((Token)retval.start).getTokenIndex(),
input.LT(-1).getTokenIndex(),
retval.st);
<endif>
>>
rewriteTemplateAlt() ::= <<
// <it.description>
<if(it.pred)>
if (<it.pred>) {
retval.st = <it.alt>;
}<\n>
<else>
{
retval.st = <it.alt>;
}<\n>
<endif>
>>
rewriteEmptyTemplate(alts) ::= <<
null;
>>
/** Invoke a template with a set of attribute name/value pairs.
* Set the value of the rule's template *after* having set
* the attributes because the rule's template might be used as
* an attribute to build a bigger template; you get a self-embedded
* template.
*/
rewriteExternalTemplate(name,args) ::= <<
templateLib.getInstanceOf("<name>"<if(args)>,
new STAttrMap()<args:{a | .put("<a.name>", <a.value>)}>
<endif>)
>>
/** expr is a string expression that says what template to load */
rewriteIndirectTemplate(expr,args) ::= <<
templateLib.getInstanceOf(<expr><if(args)>,
new STAttrMap()<args:{a | .put("<a.name>", <a.value>)}>
<endif>)
>>
/** Invoke an inline template with a set of attribute name/value pairs */
rewriteInlineTemplate(args, template) ::= <<
new StringTemplate(templateLib, "<template>"<if(args)>,
new STAttrMap()<args:{a | .put("<a.name>", <a.value>)}>
<endif>)
>>
/** plain -> {foo} action */
rewriteAction(action) ::= <<
<action>
>>
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/
/** Template subgroup to add template rewrite output
* If debugging, then you'll also get STDbg.stg loaded.
*/
group ST;
@outputFile.imports() ::= <<
<@super.imports()>
import org.stringtemplate.v4.*;
import java.util.HashMap;
>>
/** Add this to each rule's return value struct */
@returnScope.ruleReturnMembers() ::= <<
public ST st;
public Object getTemplate() { return st; }
public String toString() { return st==null?null:st.render(); }
>>
@genericParser.members() ::= <<
<@super.members()>
protected STGroup templateLib = new STGroup();
public void setTemplateLib(STGroup templateLib) {
this.templateLib = templateLib;
}
public STGroup getTemplateLib() {
return templateLib;
}
>>
/** x+=rule when output=template */
ruleRefAndListLabel(rule,label,elementIndex,args,scope) ::= <<
<ruleRef(...)>
<listLabel(elem=label+".getTemplate()",...)>
>>
rewriteTemplate(alts) ::= <<
// TEMPLATE REWRITE
<if(backtracking)>
if (<actions.(actionScope).synpredgate>) {
<alts:rewriteTemplateAlt(); separator="else ">
<if(rewriteMode)><replaceTextInLine()><endif>
}
<else>
<alts:rewriteTemplateAlt(); separator="else ">
<if(rewriteMode)><replaceTextInLine()><endif>
<endif>
>>
replaceTextInLine() ::= <<
<if(TREE_PARSER)>
((TokenRewriteStream)input.getTokenStream()).replace(
input.getTreeAdaptor().getTokenStartIndex(retval.start),
input.getTreeAdaptor().getTokenStopIndex(retval.start),
retval.st.render());
<else>
((TokenRewriteStream)input).replace(
((Token)retval.start).getTokenIndex(),
input.LT(-1).getTokenIndex(),
retval.st.render());
<endif>
>>
rewriteTemplateAlt() ::= <<
// <it.description>
<if(it.pred)>
if (<it.pred>) {
retval.st = <it.alt>;
}<\n>
<else>
{
retval.st = <it.alt>;
}<\n>
<endif>
>>
rewriteEmptyTemplate(alts) ::= <<
null;
>>
/** Invoke a template with a set of attribute name/value pairs.
* Set the value of the rule's template *after* having set
* the attributes because the rule's template might be used as
* an attribute to build a bigger template; you get a self-embedded
* template.
*/
rewriteExternalTemplate(name,args) ::= <<
templateLib.getInstanceOf("<name>")
<if(args)><args:{a | .add("<a.name>", <a.value>)}><endif>
>>
/** expr is a string expression that says what template to load */
rewriteIndirectTemplate(expr,args) ::= <<
templateLib.getInstanceOf(<expr>)
<if(args)><args:{a | .add("<a.name>", <a.value>)}><endif>
>>
/** Invoke an inline template with a set of attribute name/value pairs */
rewriteInlineTemplate(args, template) ::= <<
new ST(templateLib, "<template>")
<if(args)><args:{a | .add("<a.name>", <a.value>)}><endif>
>>
/** plain -> {foo} action */
rewriteAction(action) ::= <<
<action>
>>
/** An action has %st.attrName=expr; or %{st}.attrName=expr; */
actionSetAttribute(st,attrName,expr) ::= <<
1
2
3
4
5
6
7
8
9
10
11
12
status: 13 differences
Actions