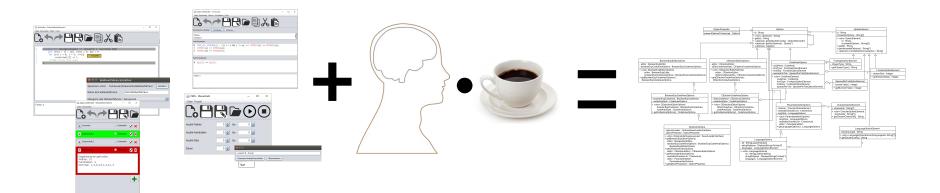


Entwurf eines Softwaresystems zur Analyse formaler Eigenschaften von Wahlverfahren

Hanselmann, Hecht, Klein, Schnell, Stapelbroek, Wohnig

INSTITUT FÜR THEORETISCHE INFORMATIK – ANWENDUNGSORIENTIERTE FORMALE VERIFIKATION



Anforderungen



- Texteditor f
 ür Beschreibung von Wahlverfahren in C
- Texteditor für Beschreibung boolscher Ausdrücke



- Editor für Parameter
- Überprüfung via CBMC







Muss- und Soll-Anforderungen – alle Module

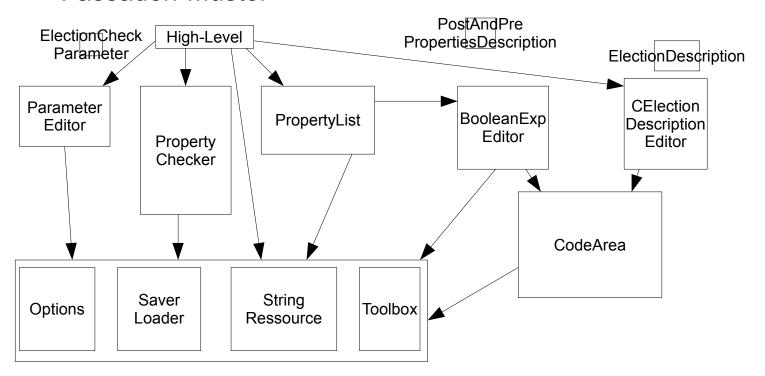


- Aufbau der Objekte bei Systemstart
- Menü, Toolbar (& Shortcuts): Eingabe → Aktion
- Speichern/Laden der Datentypen (Persistenz)
- Optionen
- Anzeigen Strings: verschiedene Sprachen
- Hängen zusammen!

Übersicht

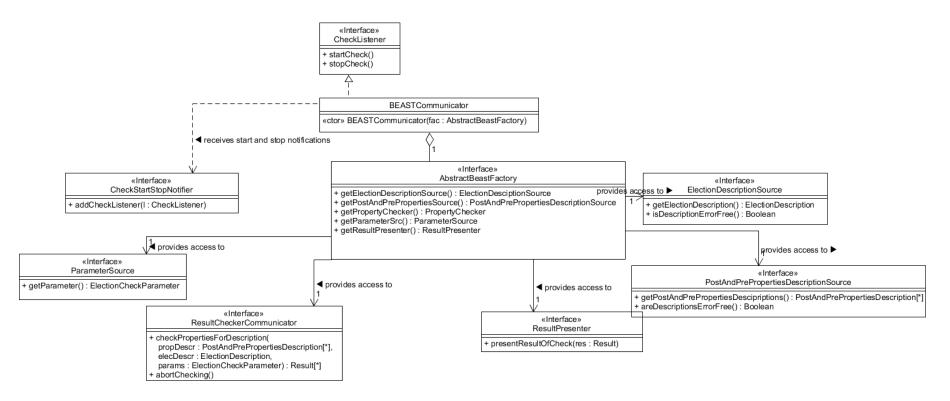


Fassaden-Muster



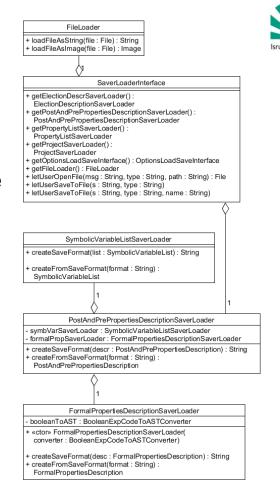
High-Level





Persistenz - SaverLoader

- Datentypen alle POD
- Modifiziertes Memento-Muster
- Pro Datentyp eine SaverLoader Klasse
 - Hierarchie der Datentypen nachvollzogen
- Zustand vom Typ String
- Alternative: Vererbung
- Vorteil: Trennung der Verantwortung

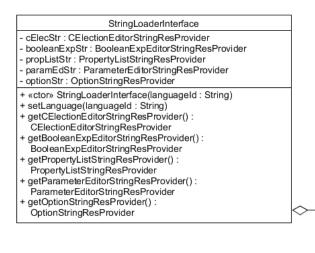


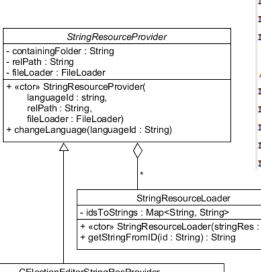


Verschiedene Sprachen – StringRessource



- Alle angezeigten Strings von Speicher geladen
- Pro Sprache ein File
- File-Format: <id>: <String>





+ getToolbarTipStringRes(): StringResourceLoader + getCErrorStringRes(): StringResourceLoader //CEditorStr_ger
menuFile: Datei
menuNew : Neu
menuSave : Speicher

menuSave : Speichern

menuSaveAs : Speichern unter

menuOpen : Öffnen

//CEditorStr_en
menuFile: File
menuNew : New
menuSave : Save

menuSaveAs : Save as

menuOpen : Öffnen

- idsToStrings : Map<String, String>
+ «ctor» StringResourceLoader(stringRes : String)
+ getStringFromID(id : String) : String

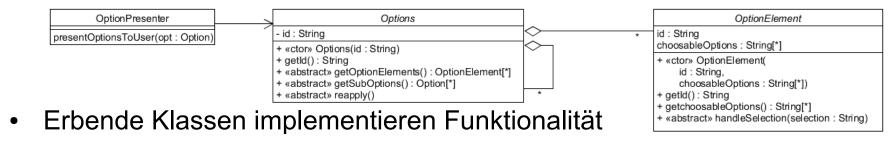
CElectionEditorStringResProvider

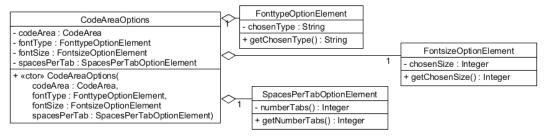
+ «ctor» CElectionEditorStringResProvider(languageId : String)
+ getMenuStringRes() : StringResourceLoader

Optionen



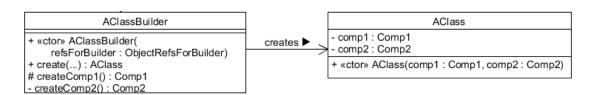
- Pro modifizierbarer Klasse eine Options-Klasse
- Präsentieren/Verändern der Optionen verallgemeinert
- Kompositums-Muster





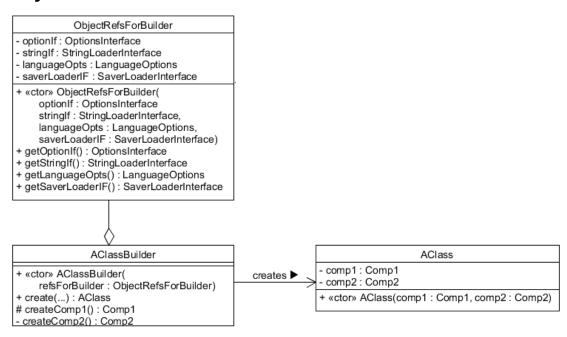


Erbauer-Muster





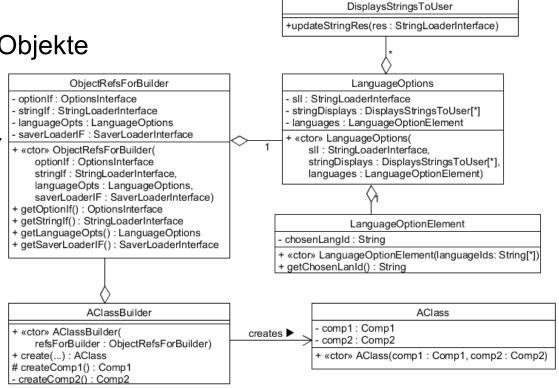
- Erbauer-Muster
- Bekommt notwendige Objekte





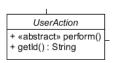
«Interface»

- Erbauer-Muster
- Bekommt notwendige Objekte
- Verlinkt
 - DisplaysStringsToUser





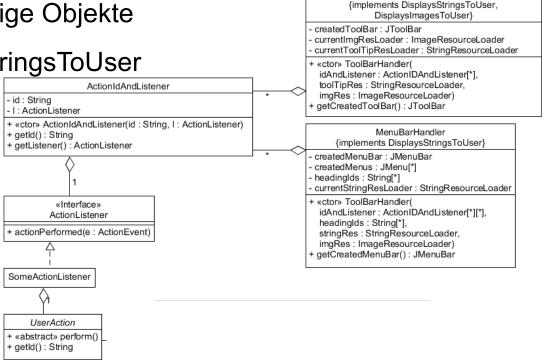
- Erbauer-Muster
- Bekommt notwendige Objekte
- Verlinkt DisplaysStringsToUser
- Erzeugt Aktionen





ToolBarHandler

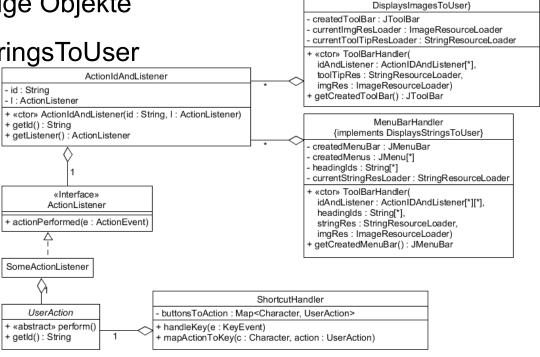
- Erbauer-Muster
- Bekommt notwendige Objekte
- Verlinkt DisplaysStringsToUser
- Erzeugt Aktionen
- Verlinkt Aktionen





ToolBarHandler {implements DisplaysStringsToUser.

- Erbauer-Muster
- Bekommt notwendige Objekte
- Verlinkt DisplaysStringsToUser
- Erzeugt Aktionen
- Verlinkt Aktionen



Aufbau von Objekten & Eingabe → Funktion &

Sprachoptionen

- Erbauer-Muster
- Bekommt notwendige Objekte
- Verlinkt DisplaysStringsToUser
- Erzeugt Aktionen
- Verlinkt Aktionen
- Pro Aktion eine Klasse



NewPropsUserAction
- editor : BooleanExpEditor
+ «ctor» NewPropsUserAction(
 editor : BooleanExpEditor,
 saveBefore : SaveBeforeChangeHandler)

SaveBeforeChangeHandler {implements UserAction, ChangeListener SavedListener, DocumentListener}

- saved : Boolean
- saver : SavePropsUserAction
- + «ctor» SaveBeforeChangeHandler(predoc : StyledDocument, postDoc : StyledDocument, varList : SymbolicVarList)

1

LoadPropsUserAction
- saverLoaderIF: SaverLoaderInterface
- editor: BooleanExpEditor

 + «ctor» LoadPropsUserAction(saverLoaderIF: SaverLoaderInterface, editor: BooleanExpEditor, saveBefore: SaveBeforeChangeHandler)

CodeArea



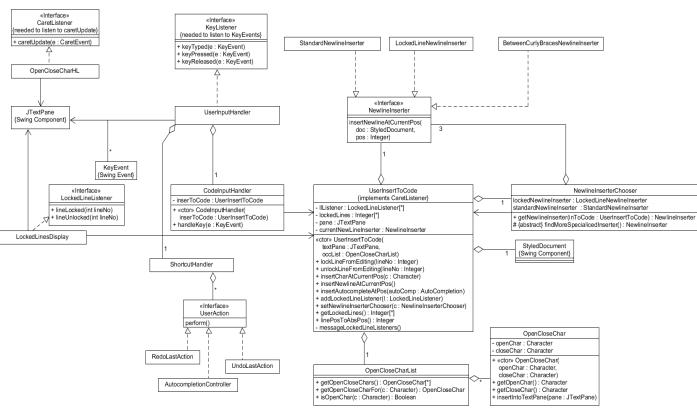
- Ein- & Ausgabe von Code
 - Schließen von Klammern etc.
- Finden & Anzeigen von Fehlern
- Rückgängig machen & Wiederholen
- Syntax Highlighting
- Erweiterbar und Spezialisierbar für Texteditoren

18.01.17

CodeArea: Ein- & Ausgabe von Code



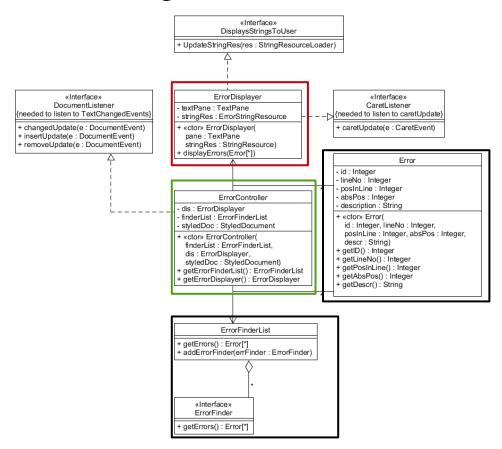
ModifiziertesMVC



CodeArea: Finden & Anzeigen von Fehlern







CodeArea: Rückgängig machen & Wiederholen



- Pro Aktion eine Klasse
- Textänderung durch TextDelta
- ActionAdder kann unterbrochen werden

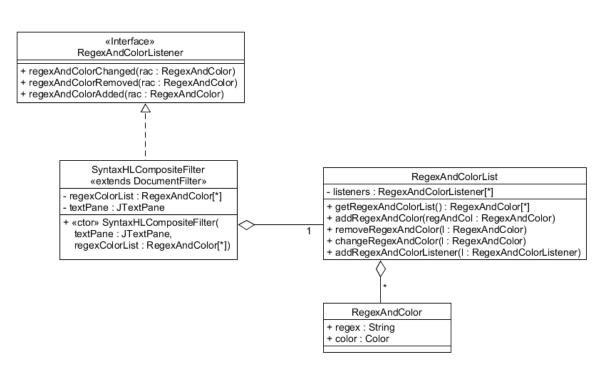


StyledDocument {Swing Component}

CodeArea: Syntax Highlighting



- DocumentFilter
- Regex und Farbe



CodeArea: Erweiterbar und Spezialisierbar für Texteditoren



Beerben von CodeArea und Implementieren von Aktionen

CodeArea # textPane : JTextPane # lineNumberDisplay : LineNumberDisplay # userInputHandler : UserInputHandler # insertToCode : UserInsertToCode # actionList : ActionList # textChangedActionHandler : TextChangedActionHandler # syntaxHL : SyntaxHL # regexAndColorList : RegexAndColorList # openCloseCharHL : OpenCloseCharHL # errorController : ErrorController # autoCompletionCtrl : AutoCompletionController «ctor» CodeArea(/*all members*/) + getCode(): String + displayCode(code : String) + getRegexAndColorList(): RegexAndColorList + getActionList(): ActionList + getErrorController() : ErrorController + getUserInputHandler(): UserInputHandler + copy() + cut() + paste() + undo() + redo()

CElectionCodeArea

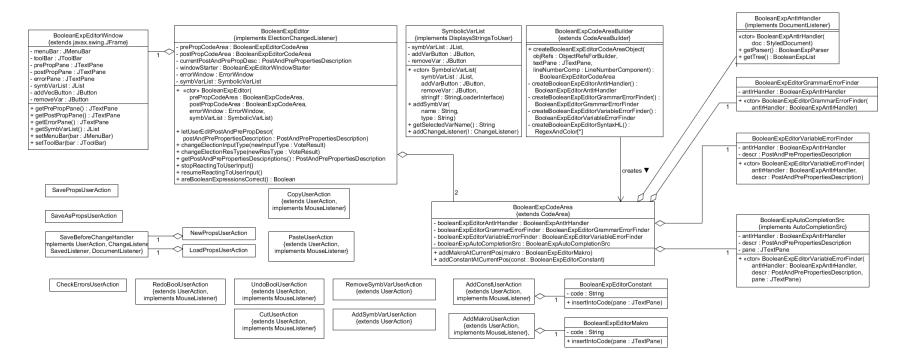
- CAntlrHandler
- CGrammerErrorFinder
- CVariableUsageErrorFinder
- CAutoCompletionSrc
- + changeElectionInputType(newInpType : VoteInput)
- + changeElectionResType(newResType : VoteResult)

BooleanExpCodeArea

- booleanExpAntlrHandler : BooleanExpAntlrHandler
- booleanExpGrammerErrorFinder : BooleanExpGrammerErrorFinder
- booleanExpVariableErrorFinder : BooleanExpVariableErrorFinder
- + changeElectionInputType(newInpType : VoteInput)
- + changeElectionResType(newResType : VoteResult)
- + addMakroAtCurrentPos(makro : BooleanExpMakro)
- + addConstantAtCurrentPos(const : BooleanExpConstant)

CodeArea: Erweiterbar und Spezialisierbar für Texteditoren





Parametereditor und Eigenschaftenliste



- Selbes Prinzip!
- Andere Interfaces & Aktionen

PropertyChecker



- Verschiedene Checker
- Verschiedene Betriebssysteme
- Mehrere Prozesse zur gleichen Zeit (?)

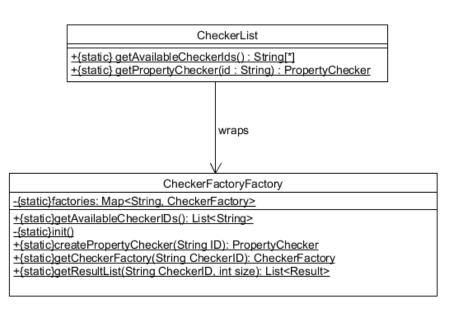
PropertyChecker: Verschiedene Checker



```
CheckerFactory
                                       {abstract}
-currentlyRunning: Checker

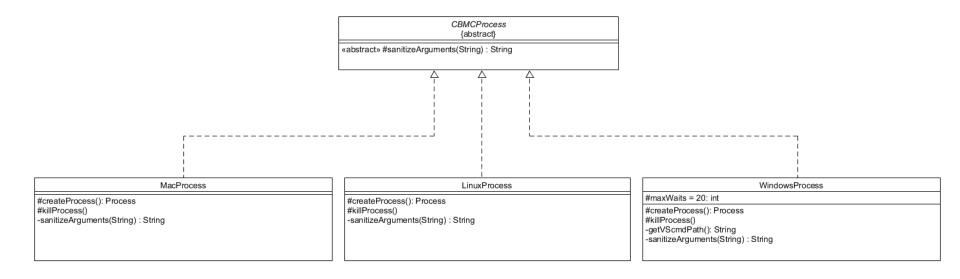
    controller: FactoryController

-workingThread: Thread
-electionDescSrc: ElectionDescriptionSrc
-postAndPrePropDescr: postAndPrePropertiesDescription
-parmSrc: ParmSrc
-result: Result
«ctor» +CheckerFactpry(
electionDescScr: ElectionDescriptionScr.
postAndPrePropDescr. postAndPrePropertiesDescription,
parmSrc: ParmSrc, result Result): CheckerFactory
+kill(): bool
«abstract» -formatInputToFile(-electionDescScr: ElectionDescriptionScr,
postAndPrePropDescr. postAndPrePropertiesDescription): String
+notifyThatFinished()
+wakeUpAndNotify()
+run()
«abstract» +getName: String
```



PropertyChecker: Verschiedene Betriebssysteme





PropertyChecker: Mehrere Prozesse zur gleichen Zeit



- Kontrolliert CheckerFactory
- Diese gibt mehrere Checker
- Diese starten einen Prozess

FactoryController

-electionDescSrc: ElectionDescriptionSrc

-postAndPrePropDescr: List<postAndPreProper</p>

-parmSrc: ParmSrc -results : List<Result>

-currentlyRunning: List<CheckerFactory>

-checkerID: int

-controllerThread: Thread -stopped = false : bool

-concurrentChecker = 4 : int

«ctor» +FactoryController(

checkerID: int,

electionDescScr: ElectionDescriptionScr,

postAndPrePropDescr: List<postAndPreProper

parmSrc: ParmSrc)

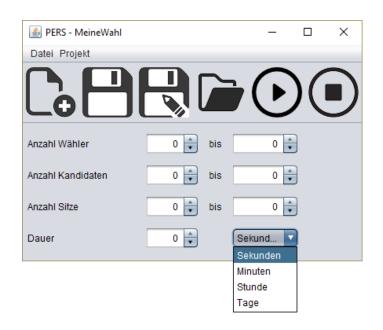
+run()

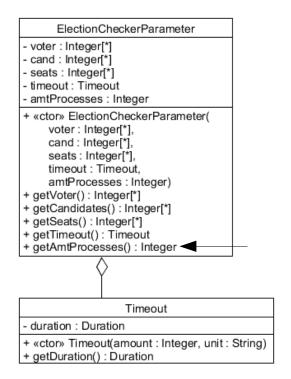
+getResults() : List<Result> +stopChecking() : bool

+getControllerThread(): Thread

Datentypen: Parameter für CBMC







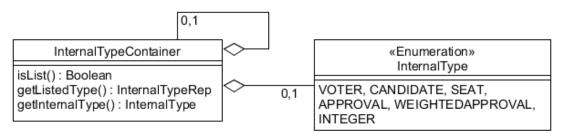
H. Klein – Entwurf 18.01.17 **28**

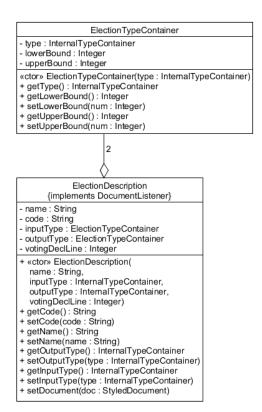
Datentypen: Beschreibung der Wahlverfahren



```
/*
 * This function describes the voting procedure
 * input: Single-Choice: Every voter picks one candidate
 * result: One candidate or a tie
 */
unsigned int voting(unsigned int votes[V]) {
    ...
}

/*
 * This function describes the voting procedure
 * input: weighted-Choice: Every voter rates every Candidate with an Integer
 * result: amount of Seats allocated per party
 */
unsigned int * voting(unsigned int votes[V][C]) {
    ...
}
```

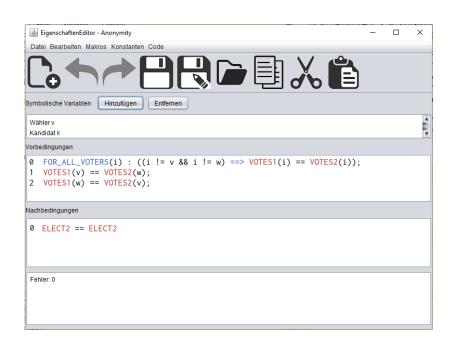


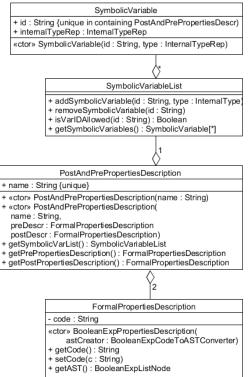


H. Klein – Entwurf 18 01 17 29

Datentypen: Beschreibung der formalen Eigenschaften

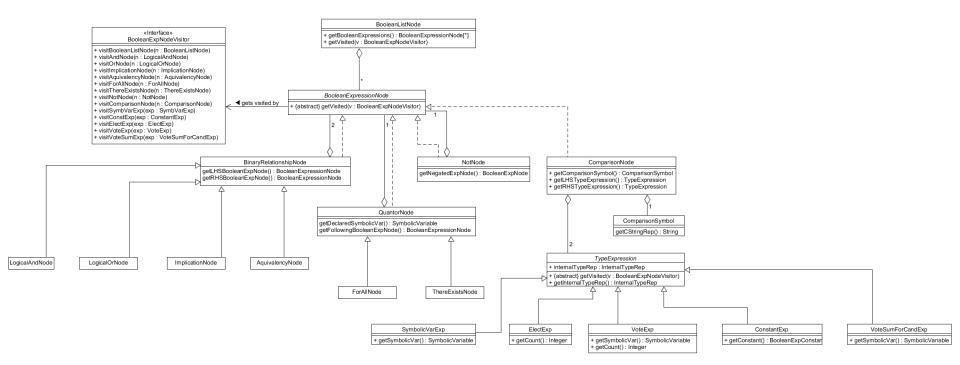






Datentypen: AST-Darstellung boolscher Ausdrücke





Erstellung des Codes für CBMC



- Eingabe: 2 * BooleanListNode + CElectionDescription
- Ergebnis: C Code für CBMC
- Implementiert BooleanExpNodeVisitor
- Vor- und Nachbedingungen analog
 - assume oder assert
- Pro BooleanExpNode eine boolsche Variable in Kellerspeicher

18.01.17

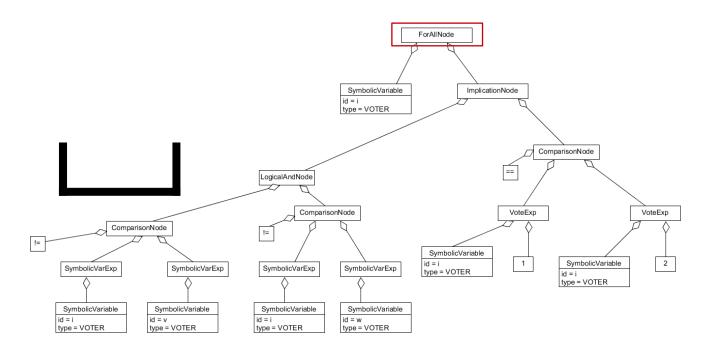
Erstellung des Codes für CBMC: Beispiel



```
* This function describes the voting procedure
                                                                                    0 FOR_ALL_VOTERS(i) : ((i != v && i != w) ==> VOTES1(i) == VOTES2(i));
* input: Single-Choice: Every voter picks one candidate
                                                                                    1 VOTES1(v) == VOTES2(w);
* result: One candidate or a tie
                                                                                    2 VOTES1(w) == VOTES2(v);
unsigned int voting (unsigned int votes[V]) {
                                                                                       ForAllNode
                                                                        SymbolicVariable
                                                                                                 ImplicationNode
                                                                      type = VOTER
                                                                                                                ComparisonNode
                                                                                                         ==
                                                     LogicalAndNode
                                                              ComparisonNode
                                                                                                          VoteExp
                                                                                                                                      VoteExp
                      ComparisonNode
                                                                                         SymbolicVariable
                                                                                                                                          2
                                                                                                                      SymbolicVariable
            SymbolicVarExp
                                   SymbolicVarExp
                                                      SymbolicVarExp
                                                                       SymbolicVarExp
                                                                                        type = VOTER
                                                                                                                     type = VOTER
            SymbolicVariable
                               SymbolicVariable
                                                     SymbolicVariable
                                                                        SymbolicVariable
                              id = v
                                                    id = i
                                                                      id = w
          type = VOTER
                              type = VOTER
                                                    type = VOTER
                                                                      type = VOTER
```

18.01.17



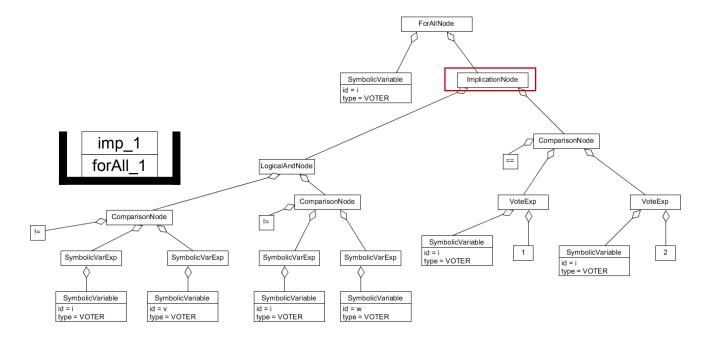




```
ForAllNode
unsigned int i;
unsigned int forAll i = 1;
for(i = 0; i < V && forAll i; i++) {</pre>
                                                                                                                                       SymbolicVariable
                                                                                                                                                                    ImplicationNode
                                                                                                                                    type = VOTER
                                                                                                                                                                                       ComparisonNode
                                                                                                                                                                              ==
                                                                     forAll_1
                                                                                                                LogicalAndNode
                                                                                                                          ComparisonNode
                                                                                                                                                                               VoteExp
                                                                                                                                                                                                                 VoteExp
                                                                          ComparisonNode
                                                                                                                 !=
                                                                                                                                                           SymbolicVariable
                                                                                                                                                                                                                      2
                                                                                                                                                          id = i
                                                                                                                                                                                  1
                                                                                                                                                                                              SymbolicVariable
                                                              SymbolicVarExp
                                                                                                                SymbolicVarExp
                                                                                          SymbolicVarExp
                                                                                                                                      SymbolicVarExp
                                                                                                                                                         type = VOTER
                                                                                                                                                                                            type = VOTER
                                                              SymbolicVariable
                                                                                                                 SymbolicVariable
                                                                                     SymbolicVariable
                                                                                                                                       SymbolicVariable
                                                                                    id = v
                                                                                                                                     id = w
                                                             type = VOTER
                                                                                    type = VOTER
                                                                                                               type = VOTER
                                                                                                                                     type = VOTER
```



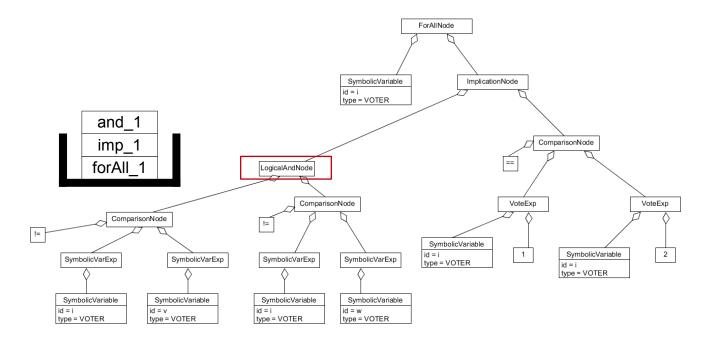
```
unsigned int i;
unsigned int forAll_i = 1;
for(i = 0; i < V && forAll_i; i++) {
    unsigned int impl 1;
```



18.01.17



```
unsigned int i;
unsigned int forAll_i = 1;
for(i = 0; i < V && forAll_i; i++) {
    unsigned int impl_1;
    unsigned int and 1;
```



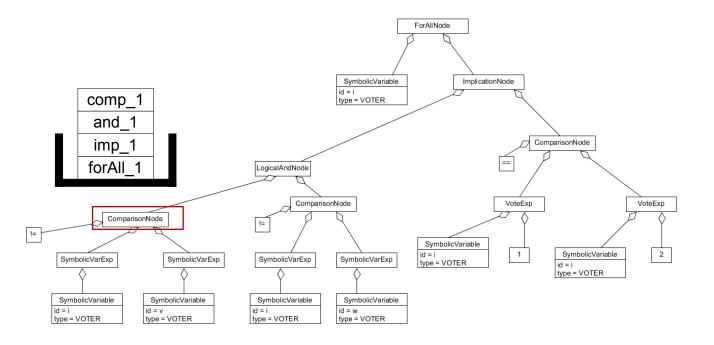


```
unsigned int i;
unsigned int forAll_i = 1;

for(i = 0; i < V && forAll_i; i++) {
    unsigned int impl_1;

    unsigned int and_1;

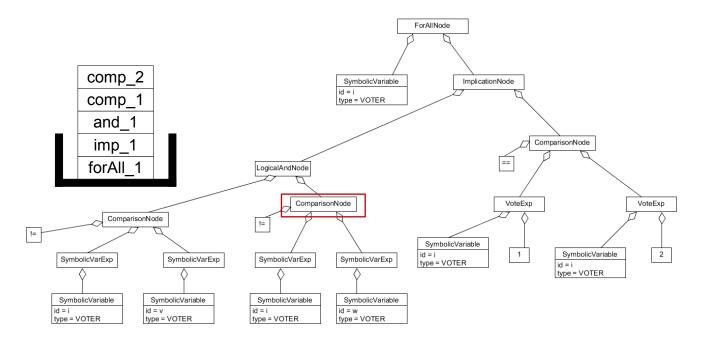
    unsigned int comp_1 = i != v;</pre>
```



18.01.17



```
unsigned int i;
unsigned int forAll_i = 1;
for(i = 0; i < V && forAll_i; i++) {
    unsigned int impl_1;
    unsigned int and_1;
    unsigned int comp_1 = i != v;
    unsigned int comp_2 = i != w;
```



18.01.17



```
unsigned int i;
unsigned int forAll_i = 1;

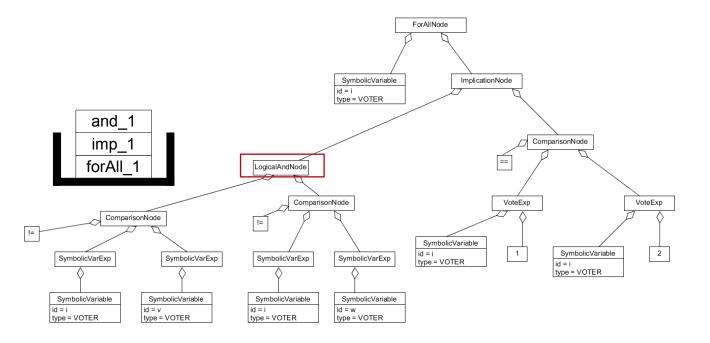
for(i = 0; i < V && forAll_i; i++) {
   unsigned int impl_1;

   unsigned int and_1;

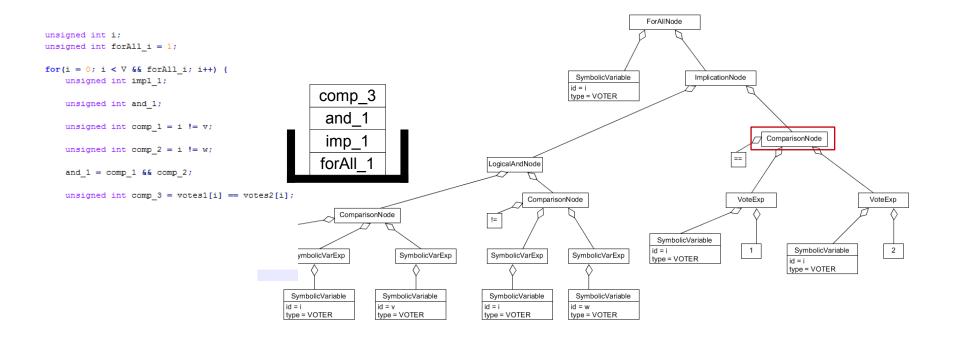
   unsigned int comp_1 = i != v;

   unsigned int comp_2 = i != w;

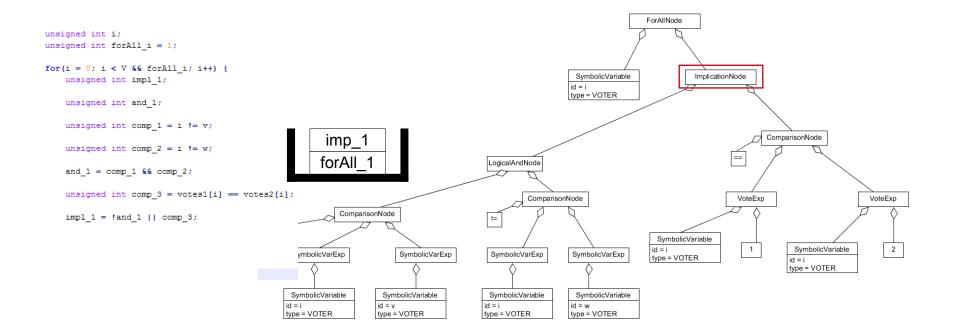
and_1 = comp_1 && comp_2;</pre>
```



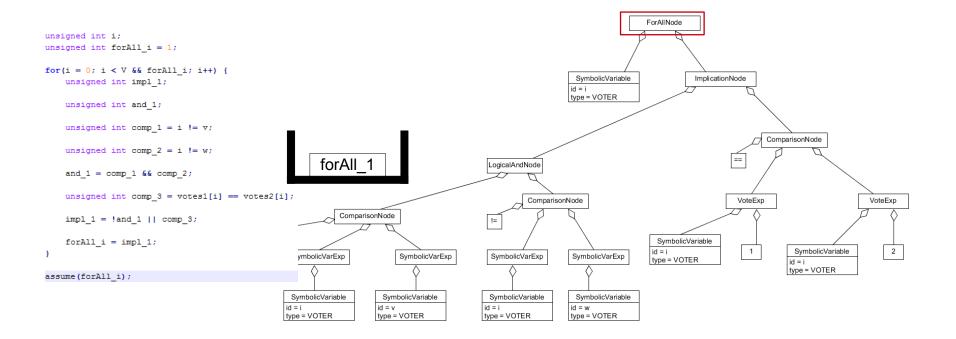














- Problem: Unteres nicht allgemein
- Was, falls Quantor && anderer Ausdruck?
- Für unser Schema kein Problem
- Demo!

```
unsigned int i:
unsigned int forAll i = 1;
for(i = 0; i < V && forAll i; i++) {
   unsigned int impl 1;
   unsigned int and 1;
   unsigned int comp 1 = i != v;
   unsigned int comp 2 = i != w;
   and 1 = comp 1 \&\& comp 2;
   unsigned int comp 3 = votes1[i] == votes2[i];
   impl 1 = !and 1 || comp 3;
   forAll i = impl 1;
assume(forAll i);
unsigned int i = 0;
for (i = 0; i < V; i++) {
    if (i != v && i != w) {
         assume (votes1[i] == votes2[i]);
```

Abweichung vom Pflichtenheft



- So gut wie keine
- + Option f
 ür Checker Auswahl
- + Option f
 ür maximal nebenläufige Checker
- + Option f
 ür Verneinung boolscher Ausdruck

18.01.17

Implementierung



