באגים בתרגום smv to core

במהלך השיפור שלי לתרגום smv to core של אייל סונסינו גיליתי כמה באגים בתרגום המקורי שאין להם תיעוד.

1. ***הדפסה של module combination –***

module combination הוא שילוב של יותר מ module instance אחד שמשולבים ביניהם בצורה סינכרונית, א-סינכרונית או קומבינציה של שילובים אלה. במבנה הנתונים של התרגום ישנו לכל TModuleInstance דגל שאומר האם המודול צריך להיות משולב סינכרונית או א-סינכרונית עם שאר המודולים. כאשר מודפסת תכנית ה core מסתכלים על הדגל הזה החל מהמודול השני והלאה בקומבינציה כך שלעובדה האם המודול הראשון צריך להיות משולב סינכרונית או לא אין השפעה על ההדפסה. במקרים בהם הדגל הזה במודול הראשון שונה מהדגל במודול השני עלולה להיווצר הדפסה שלא מתארת נכון את תכנית ה core שבמבנה הנתונים.

חשוב! באג זה קורה רק כאשר הפלט הוא במתכונתו הישנה (cdl) ולא בפלט xml שהוא ברירת המחדל של התרגום.

2. ***אי טיפול ב =! -***

התרגום הקיים אינו מזהה את סימן האי שוויון כסימן חוקי ב smv ומוציא הודעת שגיאה של סינטקס. כלי ה smv המקורי מזהה זאת כ syntax חוקי. סימן אי שוויון שיזוהה ע"י התרגום הקיים ניתן לכתוב רק בפורמט של (a = b)! .

3. ***אי טיפול בתת תחום המופיע כביטוי –***

ב smv ניתן לכתוב השמה הנותנת למשתנה ערך של תת תחום לדוגמה: x := 1..10. התרגום הקיים אינו מטפל בהשמה כזו . הוספתי הודעת שגיאה מתאימה בקובץ yacc.

עיקר הבעיה היא בתרגום הקיים מ core ל smv במקרה של relation המכיל משתנה integer או משתנה מטיפוס range. במקרה זה התכנית המתורגמת ב smv תכיל השמה של תת תחום למשתנה האמור בעוד שהתנאי של TRANS ידאג לנכונות ה relation. בכל מקרה כזה התרגום מ smv ל core לא יוכל לטפל בתכנית שתווצר כתוצאה מהתרגום של core ל smv.

פתרון אפשרי: הרחבת מבנה הנתונים של המחלקה Expression כך שביטוי יוכל להיות גם תת תחום ע"י הוספת מחלקה Subrange היורשת מ Expression. בנוסף יש לטפל במחלקה זו במחלקה TransitionBuilder כלומר לממש פונקציה:

void *TransitionBuilder::execute(SubRange *e)

שתיצור transitions מתאימים להשמה המקורית ב smv.

הוספת מחלקה חדשה יגרור שינוי ה interface של מחלקה IVisitor ושינוי דומה בכל המחלקות המממשות אותה.

4. ***DEFINE –***

ישנם כמה מקרים בהם אין טיפול ב DEFINE:

1. כאשר ישנו קבוע המוגדר ב DEFINE בתוך מודול מסוים ומסתכלים על הקבוע הזה מבחוץ למודול. בתרגום, קבוע זה צריך להיות ארגומנט של המודול שמסתכל עליו אבל ב core לא ניתן להעביר משתנה שהוגדר ב DEFINE כארגומנט למודול.
2. כאשר מגדירים ב DEFINE שדה חדש למודול שהתקבל כפרמטר (מאותה סיבה).

פתרון אפשרי: הגדרת הקבוע כמשתנה בתכנית ה core. יש לבחון האם זהו פתרון טוב בכל המקרים שיכולים להתרחש. ראה דוגמה בסוף הקובץ.

5. *מערכים רב ממדיים –*

יש טיפול חלקי במערכים רב ממדיים. ניתן להגדיר מערכים כאלה אך לא ניתן לכתוב השמה לאיבר במערך כמו a[2][4] := ... . התרגום מתעלם מהאינדקסים מלבד האינדקס הראשון ונותן אזהרה על כך.

פתרון אפשרי: על מנת לטפל במקרים אלה יש צורך לשנות ממש את מבנה הנתונים הקיים, כך שביטוי של מערך יוכל להחזיק כמה אינדקסים. יש להוסיף קוד המטפל במבנה הנתונים המורחב.

6. *דגל HOLD_PREVIOUS –*

בכל תכניות ה core שנוצרות ע"י התרגום מופיע HOLD_PREVIOUS. אייל עשה זאת על מנת לטפל ב processes של smv. התפיסה הכללית של smv היא שהערך הבא של משתנה יהיה רנדומי מעל התחום שלו אם הוא לא מצוין במפורש אך לגבי processes הדבר שונה במעט. אם ה process שפעיל כרגע אינו משנה משתנה מסוים אך process אחר שאינו פעיל כן משנה אותו, אזי לגבי משתנה זה התפיסה היא של HOLD_PREVIOUS, כלומר המשתנה שומר על ערכו הנוכחי. מתעוררת בעיה כאשר בתכנית ה smv ישנו משתנה שלא מצוינות לו השמות כלל. במקרה כזה ב smv המשתנה יקבל בכל פעם ערך רנדומי ואילו ב core בגלל ה HOLD_PREVIOUS יישאר תמיד עם ערכו ההתחלתי.

פתרון אפשרי: יש לבדוק לכל משתנה x האם יש לו השמה x := ... או next(x) := ... ואם לא יש ליצור מודול או להוסיף למודול הקיים (שנוצר בעקבות init(x)) מעבר שיבטא השמה אי דטרמיניסטית למשתנה. יכולים להיווצר מקרים שהשמה עבור משתנה מסוים לא נמצאת במודול בו מוגדר המשתנה (לדוגמה ההשמה לשדה פנימי של מודול A נעשית בתוך מודול B המקבל את A כארגומנט). צריך לחשוב מה לעשות במקרים כאלה.

An example for the “DEFINE” problem

There are some cases dealing with DEFINE in smv that Eyal doesn’t take care. The main reason for that is that in core it's impossible to send a Parameter that is originated in DEFINE as a parameter to a module, as the following core program demonstrates:

```
MODULE SYSTEM()
{
        -- if we use the first line of the following two, parse error will
        -- occur:The actual parameter a is beyond the scope of the module!
        -- The second line works properly.

        DEFINE a := 6;
--      VAR a : integer INITVAL 6;

        A(a)
}

MODULE A( b : integer)
{
        VAR
                c : integer;
        TRANS t:
        enable: true;
        assign: c':=b;

}
```