کاهش های مبتنی بر-برش برای Rebeca

چکیده

برش, یک تکنیک تجزیه و تحلیل برنامه است که می توان برای کاهش اندازه مدل و جلوگیری از انفجار حالت در چک کردن مدل استفاده نمود. در این کار, یک روش برش استاتیک برای کاهش مدل های Rebecca با توجه به یک ویژگی ارائه شده است. برای استفاده از تکنیک های برش، نمودار وابستگی Rebecca (RDG) معرفی شده است. به این دلیل که برش استاتیک معمولاً قطعه های بزرگ را تولید می کند، دو تکنیک دیگر کاهش مبتنی بر برش، برش گام به گام و برش محدود، به عنوان ایده های ساده جدید پیشنهاد شده اند. برش گام به گام تولید, قطعاتی را برای تخمین بیش از حد رفتار مدل اصلی تولید می کند و سپس آن را تصحیح می کند و برش محدود بر اساس معناشناسی تکالیف غیر-قطعی در Rebecca است. ما همچنین یک الگوریتم برش استاتیک را برای تشخیص بن بست پیشنهاد می دهیم (در غیاب هر ویژگی خاصی). قابلیت کاربرد این تکنیک ها با کاربرد آنها برای چند مطالعه موردی که در این مقاله گنجانده شده است بررسی می شود. تکنیک های مشابه را می توان در دیگر زبان های مبتنی بر-عامل اعمال نمود.

کلمات کلیدی: برش، زبان های مبتنی بر-عامل، Rebecca، چک کردن مدل، تایید، کاهش

۱. مقدمه

چک کردن مدل [۴] یک روش تأیید رسمی برای تایید سیستم های همزمان در برابر تعدادی از مشخصات است و می تواند برای توسعه سیستم های قابل اطمینان تر استفاده شود. مشکل اصلی چک کردن مدل, مشکل انفجار فضای حالت است و بسیاری از تکنیک ها برای غلبه بر این مشکل توسعه یافته اند. این روشها عبارتند از: تفسیر انتزاعی [۵]، انتزاع داده [۹]، انتزاع گزاره [۱۲]، برش [۳۱]، مرتبه جزئی [۲۳] و کاهش های تقارن [۱۵].

برای استفاده از روش چک کردن مدل، نخست باید از یک زبان مدل سازی برای نشان دادن رفتار سیستم استفاده نمود. Rebecca [۲۷] (زبان واکنشی اشیاء) یک زبان مبتنی بر-عامل با یک پایه رسمی برای مدل سازی و بررسی

سیستم های همزمان و توزیع شده است که در تلاش برای پر کردن شکاف بین روش های تأیید رسمی و برنامه های کاربردی واقعی طراحی شده است. در [۲۶], اجزا برای زبان Rebecca به منظور بسته بندی محکم اشیاء واکنشی جفت شده معرفی شده اند. این زبان با مجموعه ای از ابزار های چک کردن مدل [۱۷،۲۸،۲۹] پشتیبانی می شود.

برش استاتیک [۳۱], دستوراتی را از یک برنامه گرفته است که دارای اثر مستقیم یا غیر مستقیم بر محاسبات خاص است. یکی از رویکردهای اصلی برای برش, استفاده از تجزیه و تحلیل قابلیت دسترسی در نمودار وابستگی برنامه است.

برای مدل های برش Rebecca, یک گراف وابستگی باید ابتدائاً ساخته شود. برای این منظور, ما گراف وابستگی خاص را بر اساس معناشناسی Rebecca معرفی می کنیم. این نمودار دارای پیچیدگی کمتر از نمودارهای وابستگی موجود است که دلیل آن, ماهیت ناهمزمان ارتباطات، اجرای اتمی سرورهای پیام، عدم وجود متغیرهای به اشتراک گذاشته و عدم تماس های رویه است (از این رو هیچ نیازی به لبه های تداخل و یا خلاصه مورد بحث در [۱۹] وجود ندارد) .

علاوه بر این، هر چند Rebecca, یک زبان مبتنی بر-شی است، نباید با پیچیدگی های نمودارهای وابستگی طراحی شده برای زبان های شی گرا سرو کار داشته باشیم، زیرا ویژگی هایی مانند وراثت و چند ریختی در زبان گنجانده نشده اند. در مورد مدل های مبتنی بر جزء, زیرگراف مربوطه از هر یک از مؤلفه ها را می توان ذخیره نمود و زمانی که یک جزء در مدل دیگری به نظر می رسد, مورد استفاده مجدد قرار داد.

برای محاسبه برش از نمودار حاصل، چهار الگوریتم مختلف در این مقاله ارائه شده است. اولین الگوریتم, الگوریتم قابلیت دسترسی سنتی است که برای برش استاتیک استفاده می شود. الگوریتم دوم بر اساس یک ایده ساده بر اساس جدید است و زمانی استفاده می شود که ما می خواهیم یک مدل را در برابر بن بست چک کنیم (بر خلاف الگوریتم های برش منظم، هیچ نیازی به مشخص نمودن یک ویژگی در اینجا وجود ندارد). ایده کار, حذف دستوراتی است که هیچ تاثیری در هیچ بیانیه ای دیگری ندارند.

در الگوریتم برش سوم، برش گام به گام، یک تخمین بیش از حد از مدل اصلی محاسبه می شود و سپس بر اساس نتیجه تایید، مدل کاهش یافته در صورت نیاز اصلاح می شود. این الگوریتم توسط گنجاندن متغیرهای ویژگی در

مدل شروع می شود. متغیرهایی که دارای اثر مستقیم بر ارزش متغیرهای ویژگی هستند نیز در مدل گنجانده می شوند. این متغیرها یک مقدار را با استفاده از یک انتساب غیر قطعی در مدل کاهش یافته می گیرند. متغیرهای دیگر از مدل حذف می شوند. سپس, مدل کاهش یافته تایید می شود و در صورتی که یک مثال نقض جعلی پیدا شود، مدل با گنجاندن متغیرهای بیشتری در آن اصلاح می شود.

الگوریتم گذشته، به نام برش محدود را می توان به عنوان یک رویکرد میانی بین برش استاتیک و برش گام به گام در نظر گرفت. برش استاتیک, ویژگی را به شدت حفظ می کند اما برش های بزرگ از جمله بسیاری از متغیرها را تولید می کند. از سوی دیگر، برش گام به گام فقط شامل چند متغیر معدود در مدل کاهش یافته در گام اول می شود، اما مدل را بیش از حد تخمین می زند و ممکن است به چند مرحله اصلاح نیاز داشته باشد. در برش محدود، الگوریتم برش استاتیک توسط دستورات تکالیف غیر قطعی محدود می شود. دلیل آن این است که هیچ بیانیه ای در این برنامه وجود ندارد که احتمالاً بتواند ارزش این تکالیف را تحت تاثیر قرار دهد. کاربر می تواند فرایند برش را با ارائه متغیرهای بیشتر به الگوریتم برش محدود, محدودتر نماید. اینها متغیرهایی هستند که ارزش واقعی آنها در هنگام چک کردن یک ویژگی خاص، بر اساس اطلاعات کاربر مهم نیست. الگوریتم برش محدود, جایگزین تکالیف واقعی می شود که یک مقدار را به این متغیرها با تکالیف غیرقطعی منسوب می کند و متغیرهای دیگر مؤثر بر ارزش این متغیرها را حذف می کند.

اگرچه مدل کاهش یافته, رفتار مدل اصلی را بیش از حد تخمین می زند، اما احتمال یافتن مثال نقض جعلی کاهش می یابد. دلیلش این است که متغیرها به طور ابتکاری توسط کاربر حذف می شوند (و به عنوان یک شیوه موقت). با این حال در مورد پیدا کردن یک مثال نقض جعلی, این مدل باید با اضافه کردن متغیرهای بیشتر به آن اصلاح شود.

سهم این مقاله, معرفی تکنیک های برش Rebeca است. تکنیک های کاهش دردسترس برای Rebecca, کاهش تقارن [۱۸] و تایید ترکیبی [۲۸،۲۹] هستند. مزایای اضافه کردن تکنیک های برش به تکنیک های کاهش موجود برای Rebecca عبارتند از:

• ترکیب با تکنیک های کاهش دیگر: برش را می توان در ترکیب با تکنیک های کاهش دیگر از جمله تایید ترکیبی و کاهش تقارن و مدلسازی مدل های بزرگتر چک کردن استفاده نمود.

• پردازش خودکار: فرآیند برش استاتیک کاملاً خودکار است و به کاربر در فرآیند کاهش مرتبط نمی شود, نسبت به رویکرد تایید ترکیبی که در آن کاربر باید در انتخاب تعداد قطعات تصمیم بگیرد. برش محدود به طور خودکار بر روی یک مدل Rebecca اعمال می شود که از انتساب غیر قطعی برای انتساب مقدار به برخی از متغیرها استفاده می کند. با این حال کاربر می تواند متغیرهای بیشتر (با مقادیر غیر قطعی) را برای الگوریتم برش محدود برای رسیدن به یک یک تکه کوچکتر مشخص نماید. برش گام به گام به طور کامل در این کار به صورت خودکار عمل نمی کند چرا که فرآیند اصلاح نیاز به تعامل با کاربر دارد. اما می توان آن را به یک فرایند کاملا خودکار بهبود داد و این یکی از کارهای آینده است.

• حفظ ویژگی: برش استاتیک توسط حفظ قوی ویژگی است. این بدان معنی است که رضایت و نقض یک ویژگی در مدل اصلی را می توان به طور مستقیم از مدل کاهش یافته استنتاج نمود. در مقابل، تایید ترکیبی, مدل را بیش از حد تخمین می زند و نقض ویژگی در مدل کاهش یافته لزوماً حاکی از نقض ویژگی در مدل اصلی نیست. هر دوی روش برش گام به گام و برش محدود مدل را بیش از حد تخمین می زنند، با این حال زمانی که برش محدود استفاده می شود، امکان پیدا کردن مثال نقض جعلی کاهش می یابد.

موارد جدید در روش ما را می توان به صورت زیر خلاصه نمود:

• معرفی یک گراف وابستگی ویژه برای Rebecca با توجه به معانی زبان مبتنی بر-عامل، که پیچیدگی های نمودارهای موجود را ندارد. این نمودار را می توان برای مدل های Rebecca مبتنی بر جزء اعمال نمود، و در این مورد زیرگراف یک جزء را می توان برای استفاده مجدد بیشتر ذخیره نمود.

• ارائه یک روش برش برای مدل های برش که باید در برابر بن بست ها تأیید شوند (نه یک ویژگی خاص).

• ارائه یک روش برش به نام برش گام به گام ، که به واسطه تخمین بیش از حد رفتار یک مدل Rebecca, موجب تولید قطعات کوچکتر می شود.

• ارائه یک روش برش به نام برش محدود، بر اساس انتساب غیر قطعی به متغیرها در Rebecca.

تکنیک های مشابه (از جمله نمودار و الگوریتم های وابستگی) را می توان برای زبان های مبتنی بر-عامل مشابه بکار گرفت. علاوه بر این، این تکنیک ها را می توان در ترکیب با تکنیک های کاهش دیگر استفاده نمود.

این مقاله به شرح زیر سازماندهی شده است. در بخش بعدی, یک نمای کلی از کارهای مرتبط ارائه شده است. بخش ۳, خلاصه معرفی روش برش زبان و برنامه Rebecca. در بخش ۴ نمودار وابستگی Rebecca ارائه شده است و در بخش ۵ الگوریتم برش های مختلف مورد بحث قرار گرفته است. بخش ۶ نتیجه استفاده از روش برش برای دو مطالعه موردی را توضیح می دهد و بخش آخر, نتیجه کار را توضیح می دهد.

۲ کار مرتبط

برش استاتیک به عنوان یک روش کاهش در [۱،۶،۲۱،۱۳،۲،۲۴] برای مقاصد چک کردن مدل استفاده می شود. در [۷] یک ارزیابی از استفاده از این روش برای کاهش مدل ارائه شده است. نتیجه [۷] نشان می دهد که کد منبع شی گرای برش همزمان, کاهش های قابل توجهی را فراهم می کند که متعامد بر تعداد تکنیک های کاهش دیگر است و برش همیشه باید با توجه به اتوماسیون و هزینه محاسباتی کم استفاده شود.

یک رویکرد به نام برش انتزاعی، در [۱۴] ارائه شده است که بر اساس تفسیر انتزاعی است. برش انتزاعی (چکیده) برش استاتیک را با گزاره ها و محدودیت ها با استفاده از مدل برنامه به عنوان نمودار حالت انتزاعی گسترش می دهد که با استفاده از انتزاع گزاره برای یک برنامه به دست می آید. برای کنترل مشکل انفجار فضای حالت، برش انتزاعی از نظر چک کردن نمادین مدل فرموله می شود. در این تکنیک انتزاع، می توان تعیین نمود که تحت کدام شرایط یک دستور می تواند دیگری را تحت تاثیر قرار دهد. اما برای تأیید ما ممکن است نیاز به پیدا کردن این مورد داشته باشیم که آیا برخی از شرایط ممکن است همیشه برقرار باشند یا هرگز برقرار نباشند.

یکی از ایده های به تازگی ارائه شده, برش افزایشی [۳۰] است. این کار با یک بخش کوچک و حداقل مشخصات شروع می شود و پی در پی قطعات بیشتر را می افزاید تا زمانی که هر ویژگی تحت نظر روی یک قطعه برقرار باشد یا یک مثال نقض واقعی پیدا شود. این تکنیک برای CSP-OZ [10] استفاده می شود. روش برش گام به گام ارائه شده

در این مقاله, از ایده تخمین بیش از حد رفتار مدل و سپس اصلاح آن استفاده می کند. با این حال به دلیل ماهیت متفاوت زبان ها, روش استفاده از ایده ها, متفاوت است. علاوه بر این، در [۳۰] این تکنیک برای یک دستگاه ساده استفاده می شود (در مقایسه با کار ما که در آن این تکنیک برای گراف وابستگی اعمال می شود) و مقایسه بیشتر بین این دو تکنیک امکان پذیر نیست.

در [۱۱،۲۰] یک تکنیک است برای برش سیستم های واکنشگر همزمان با معرفی یک مفهوم جدیدی از برش ارائه شده است. در [۱۱]، این روش را به زبان Argos در است که در ماشین متناهی بر اساس اعمال می شود. در [۲۰] زبان Esterl، در نظر گرفته شده است که مجموعه ای غنی از الگوهای کنترل را دارد. تمرکز [۲۰] روی مدل سازی این الگو ها با تعریف وابستگی های جدید است. تفاوت اصلی کار ما و این روش, ماهیت ناهمزمان و مبتنی بر عامل زبان Rebecca است.

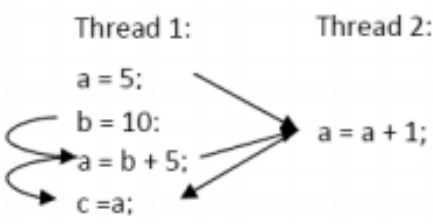
```
reactiveclass Sender(2){
   knownrebecs{
      Receiver receiver;
   }
   statevars{
      int x;
      boolean y;
   }
   msgsrv initial(){
      x = 0;
      y = false;
      self.dataSend();
   }
   msgsrv dataSend(){
      if(y)
         x = x + 1;
      receiver.dataReceive(x);
      if(x == 5)
         x = 0;
      y = ?(true,false)
   }
}

main {
   Sender sender(receiver):();
   Receiver receiver(sender):();
}
```

```
reactiveclass Receiver(2){
   knownrebecs{
      Sender sender;
   }
   statevars {
      int a;
      boolean b;
   }
   msgsrv initial(){
      a = 0;
      b = false;

   }
   msgsrv dataReceive(int msg){
      a = msg;
      if(a == 5)
          b = true;
      else
         b = false;
      sender.dataSend();
   }
}
```

شکل ۱. نمونه ای از مدل ربکا



شکل ۲. نمونه ای از یک قطعه نادرست در برنامه های همزمان

۳ مقدمات

۳,۱ Rebecca

Rebecca [۲۷] یک زبان مبتنی بر-عامل برای مدل سازی سیستم های همزمان و توزیع شده به عنوان یک مجموعه ای از اشیاء واکنشی است که از طریق عبور پیام ناهمزمان ارتباط برقرار می کنند. یک مدل Rebecca متشکل از مجموعه ای از رده های واکنشی است. هر کلاس واکنش شامل مجموعه ای از متغیرهای حالت و مجموعه ای از سرور های پیام می شود که در آن بدنه سرورهای پیام به صورت اتمی اجرا می شود. در یک مدل Rebecca, مجموعه ای از rebecs (اشیاء واکنشی) وجود دارند که به صورت همزمان اجرا می شوند. Rebecs, اشیاء محصور واکنشی بدون متغیرهای به اشتراک گذاشته هستند. هر rebec از یک کلاس واکنشی نمونه برداری می شود و دارای یک موضوع واحد اجرا است که توسط خواندن پیام ها از صف بیکران تحریک می شود. هر پیام, یک روش منحصر به فرد را مشخص می کند که هنگامی که پیام سرویس دهی می شود, باید درخواست شود. هنگامی که یک پیغام از صف خوانده می شود، روش آن درخواست می شود و پیام از صف حذف می شود. هر rebec دارای یک سرور پیام اولیه است و در حالت اولیه, صف rebec خالی است و دستور اجرایی آن دستور اول سرور پیام اولیه است.

در [۲۶]، اجزا, اشیاء واکنشی جفت شده محکم را که می توانند ارتباط همزمان داشته باشند بسته بندی می کنند. رفتار هر جزء, مانند یک شی واکنشی است و در ساده ترین حالت, هر شی واکنشی به خود خود یک جزء است. در این مقاله ما از ارتباط همزمان داخلی استفاده می کنیم, زیرا این رفتار طبیعی برای عاملان نیست.

شکل ۱ یک مثال بسیار ساده Rebecca برای نشان دادن دستور و معناشناسی Rebecca و تکنیک های برش ما است. این مثال مشابه با پروتکل بیت متناوب است، اما ما آن را با قرار دادن یک انتساب غیر قطعی به جای دریافت یک اذعان واقعی توسط فرستنده ساده سازی نموده ایم.

References

[1] Bozga, M., J.-C. Fernandez, L. Ghirvu, S. Graf, J.-P. Krimm and L. Mounier, *If: An intermediate representation and validation environment for timed asynchronous systems*, World Congress on Formal Methods (1999).

[2] Bruckner, I. and H. Wehrheim, *Slicing an integrated formal method for verification*, In ICFEM 2005: Seventh International Conference on Formal Engineering Methods, volume 3785 of LNCS (2005), pp. 360–374.

[3] Cheng, J., *Slicing concurrent programs–a graph-theoretical approach*, Proceedings of the First International Workshop on Automated and Algorithmic Debugging **749** (1993), pp. 223–240.

[4] Clarke, E. M., O. Grumberg and D. A. Peled, "Model Checking," The MIT Press, 2000.

[5] Cousot, P. and R. Cousot, *Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints*, In Proceedings of POPL77 (1977), pp. 238–252.

[6] Dwyer, M. B., J. Hatcliff, M. Hoosier, V. Ranganath, Robby and T. Wallentine, *Evaluating the effectiveness of slicing for model reduction of concurrent object-oriented programs*, TACAS (2006).

[7] Dwyer, M. B., J. Hatcliff, M. Hoosier, V. Ranganath, Robby and T. Wallentine, *Evaluating the effectiveness of slicing for model reduction of concurrent object-oriented programs*, TACAS (H. Hermanns, J. Palsberg,Eds.), 3920, Springer (2006).

[8] E. M. Clarke, S. J. Y. L., O. Grumberg and H. Veith, *Counterexample-guided abstraction refinement for symbolic model checking*, JACM (2003), pp. 752–794.

[9] E.M. Clarke, O. G. and D. Long, *Model checking and abstraction*, In 19th ACM POPL (1992).

[10] Fischer, C., *CSP-OZ: A combination of Object-Z and CSP*, In H. Bowman and J. Derrick, editors, Formal Methods for Open Object-Based Distributed Systems(FMOODS 97) **2** (1997), pp. 423–438.

[11] Ganapathy, V. and S. Ramesh, *Slicing synchronous reactive programs*, Electronic Notes in Theoretical Computer (2002).

[12] Graf, S. and H. Saidi, *Construction of abstract state graphs with pvs*, In Proceedings of CAV97 (1997).

[13] Hatcliff, J., M. B. Dwyer and H. Zheng, *Slicing software for model construction*, Higher-Order and Symbolic Computation (2000), pp. 315–353.

[14] Hong, H., I. Lee and O. Sokolsky, *Abstract slicing: A new approach to program slicing based on abstract interpretation and model checking*, SCAM, IEEE Computer Society (2005).

[15] Ip, C. and D. Dill, *Better verification through symmetry*, In International Conferenceon Computer Hardware Description Languages (1993).

[16] J. Ferrante, K. J. O. and J. D. Warren, *The program dependence graph and its use in optimization*, ACM Trans.Prog. Lang. Syst (1987), pp. 319–349.

[17] Jaghoori, M., A. Movaghar and M. Sirjani, *Modere: The model-checking engine of Rebeca*, ACM Symposium on Applied Computing - Software Verification Track (2006), pp. 1810–1815.

[18] Jaghoori, M. M., M. Sirjani, M. R. Mousavi and A. Movaghar, *Efficient symmetry reduction for an actor-based model*, ICDCIT **LNCS 3816** (2005), pp. 494–507.

[19] Krinke, J., *Context sensitive slicing of concurrent programs*, ACM SIGSOFT Software Engineering Notes (2003).

[20] Kulkarni, A. R. and S. Ramesh, *Static slicing of reactive programs*, Source Code Analysis and Manipulation, SCAM (2003).

[21] Millett, L. and T. Teitelbaum, *Issues in slicing promela and its applications to model checking, protocol understanding, and simulation*, Software Tools for Technology Transfer (2000), pp. 343–349.

[22] Nanda, M. and S. Ramesh, *Slicing concurrent programs*, Software Engineering Notes (2000), pp. 180–190.

[23] Peled, D., *All from one, one for all: On model checking using representatives*, In Proceedings 5th Workshop on Computer Aided Verification, number 697 (1993).

[24] Qi, X. and B. Xu, *An approach to slicing concurrent ada programs based on program reachability graphs*, IJCSNS International Journal of Computer Science and Network Security (2006).

[25] S. Horwitz, T. R. and D. Binkley, *Interprocedural slicing using dependence graphs*, ACM Transactions on Programming Languages and Systems (1990), pp. 26–61.

[26] Sirjani, M., F. de Boer and A. Movaghar, *Modular verification of a component-based actor language*, Journal of Universal Computer Science **11** (2005), pp. 1695–1717.

[27] Sirjani, M., A. Movaghar, A. Shali and F. de Boer, *Modeling and verification of reactive systems using Rebeca*, Fundamenta Informaticae **63** (2004), pp. 385–410.

[28] Sirjani, M., A. Movaghar, A. Shali and F. S. de Boer, *Model checking, automated abstraction, and compositional verification of Rebeca models*, J.UCS 11(6) (2005), pp. 1054–1082.

[29] Sirjani, M., A. Shali, M. Jaghoori, H. Iravanchi and A. Movaghar, *A front-end tool for automated abstraction and modular verification of actor-based models*, In Proceedings of ACSD **63** (2004), pp. 145–148. IEEE Computer Society.

[30] Wehrheim, H., *Incremental slicing*, ICFEM 2006 (2006), pp. 514–528.

[31] Weiser, M., *Program slicing*, In Proceedings of the 5th international conference on Software engineering (1981), pp. 439–449.