Mühlenpfordtstr. 23
38106 Braunschweig
Phone: 0531 - 391 9524
Email: roland.meyer@tu-bs.de

July 7, 2024

Report on the PhD thesis of Petr Janku

String constraints formulate requirements on strings, like the equality between strings, constraints on the length, membership requirements in regular languages, and constraints that should hold after a transformation of the strings. String constraints are fundamental in a variety of settings, from the automatic trading between companies to text processing, and of course in the context of web applications, which is the example that I will use in this report. Consider the correctness condition that every input to the web application is sanitized before processing. This can be expressed as the fact that every string to be processed is the result of a suitable transformation. The example shows that string constraints are fundamental to describing the requirements on web-based systems. But they are also fundamental to proving the correctness resp. to finding bugs. When proving correctness, the program is not only annotated by a correctness condition as stated above, but one also annotates (to a large degree automatically) every line of code by an assertions formulated as a string constraint. The correctness condition then holds if every string constraint entails the next one. This method for proving correctness is called deductive verification or verification condition checking. When hunting bugs instead of proving correctness, there is a dynamic and a static approach. In the static approach, one unwinds the system for a number of steps and encodes this unwinding into a string constraint. The string constraint expresses the fact that these steps lead to a violation of the costraint. The dynamic approach is less eager but only builds the string constraint along a single execution.

Contributions In all settings described above, proving correctness of the web application resp. finding a bug amounts to solving a string constraint. The contribution of Petr Janku's thesis are new classes of string constraints that are useful to capture verification tasks that occur in practice, and efficient algorithms for solving these constraints. Given the **relevance** of string-based applicationas as explained above, and the crucial role that string constraints play in their verification, I consider the thesis highly relevant and extremely timely.

There are four contributions that I will now elaborate on.

The first constribution is a new algorithm for solving string constraints that in particular support concatenation and string transformations expressed by finite-state transducers. Note that this fragment is already enough to express the above requirement for sanitization. The key insight is that the solution space of these constraints can be adequately captured by an alternating finite automaton (AFA). This means the existence of a solution to the constraints amounts to non-emptiness of the language of the automaton. The challenge is to keep the AFA small when string transformations come into play. The solution proposed by Janku consists of two steps. First, he shows that a compact encoding can be achieved with a symbolic alphabet. Second, he gives efficient algorithms for solving AFA over symbolic alphabets. The solution has been implemented and shows an excellent performance.

The second contribution is the definition of a new fragment of string constraints that admit an (again new and) efficient solving algorithm. It is the so far largest fragment of string constraints that can be handled algorithmically. The problem with the fragment which was previously known to be decidable is that it forbids the comparison of variables when they go through different transformations. So-called chain free constraints admit such a comparison as long as the constraints do not form a cycle. The technical notion is that of a chain and defined through a cleverly constructed graph that can be efficiently constructed from the string constraint. The new algorithm for solving chain-free constraints is highly non-trivial and consists of several steps. To name two highlights, the steps include the elimination of chains and concatenation, and the computation of Parikh images.

The third contribution is a new algorithm for solving string constraints that in particular have conversions of strings to numbers (as relevant in web applications). It should be noted that the problem is undecidable in general. The algorithm is therefore approximative, but has the ability to refine its precision on-the-fly. The crucial idea is to encode strings that are supposed to represent numbers by a cleverly designed new automaton model, so-called parametric flat automata (PFA). Parametric refers to the fact that the precision of the approximation can be refined. Flat means the automata do not have nested loops. With PFA, it is possible to give a polynomial-time encoding of the approximation into linear constraints.

The last contribution builds on the idea of solving string constraints with approximate methods. The approximation in this case is via Parikh images that are computed from an AFA. The algorithm behind this is new, and generalizes a famous result due to Verma, Seidl, and Schwentick in an unexpected way.

Judgment The contributions meet highest standards in terms of **significance** and **novelty**. In terms of significance, please refer to the introductory paragraphs given above. Let me elaborate on the novelty. Petr Janku links string constraints to verification algorithms by going through normalization and automata models, namely AFA with symbolic alphabets and PFA. By following this path, Petr Janku demonstrated a number of abilities. First, he demonstrates firm knowledge in several and rather distinct fields, namely string constraints, logic, automata, and verification. Second, it is not at all clear that string constraints and verification algorithms are a good match, and it requires creativity to come up with the idea of linking the two. Third, even if one has the idea, it still requires technical mastership to define normalization algorithms and suitable automata models to put the idea into practice. Petr Janku has demonstrated all this, in an impressive line of work.

As a consequence of this successful line of work, Janku's contributions have been **published** in conferences of highest rank, including POPL (Conference Rank A*, Contribution 1), PLDI (Conference Rank A* Contribution 3), ATVA (Conference Rank A, Contribution 2), and EURO-CAST (Conference Rank B, Contribution 4).

Also the **formal aspects** of the thesis are met with high quality. The explanations are given at the right level of abstraction. The definitions that are needed to state the formal results are given, technical details, however, are relegated to the appendix. This choice allowed me to go through the thesis rather quickly and immediately grasp the ideas. The many helpful illustrations and examples that are given all along the way certainly helped.

The **scientific activities** clearly indicate that Petr Janku is a person with scientific erudition and creative abilities. The high-class publication list and collaborations with stars in our field underline this impression.

In my opinion, the thesis and the student´s achievements until now meet the generally accepted requirements for

the award of an academic degree

(in accordance with Section 47 of Act No. 111/1998 Coll., on higher education institution).

Best regards

Prof. Dr. Roland Meyer