The Name of the Title is Hope

YANJUN CHEN**, University of Michigan, USA

ACM Reference Format:
Yanjun Chen. 2021. The Name of the Title is Hope. 1, 1 (June 2021), 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Stepper that can display all immediate states of a program is a useful tool for debugging and education. Findler et al.[citation] developed a stepper for DrScheme in their programming environment. Cong and Asai [citation] used the contextual dynamics to build a stepper, which takes subexpression out and plug in after evaluation. In this paper, we will use the similar strategy to develop the stepper.

In Hazel programming language, the incomplete programs are also allowed for evaluation. It contains an extra variable called hole indicating the place that needs to fill. There are two examples shown in Figure 1.

[Paused example]

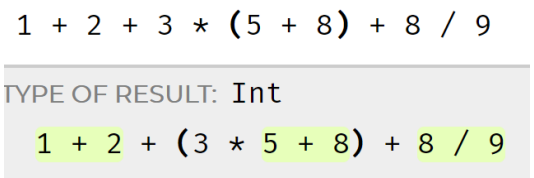

Fig. 1. Example of Multiple Environment

In the first example, the expression is already simple enough and it is unnecessary to expand cases since the parameter has not provided. Therefore, stepper need to detect this situation so that it will pause until user requires it to progress. Another example has an expression with two parts. The left part is heavy since it takes about 16 steps to final while user might just want to debug the right part. According to the regular dynamics, there should contain only one way to step for any non-final programs. So, user has to click 16 times to reduce the left one in order to debug the right one. The possible solution is just allowing the multiple evaluation contexts so that user can choose the subexpression they need. This paper develops a stepper with multiple evaluation contexts and pausing environment.

*Research advisor: Cyrus Omar

Author's address: Yanjun Chen, yanjunc@umich.edu, University of Michigan, USA.

XXXX-XXXX/2021/6-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 CONTEXTUAL DYNAMICS

The syntax of DH-types and DH-expression is shown in Figure 2. We use the simplified definition given in [Omar] combined with number and addition.

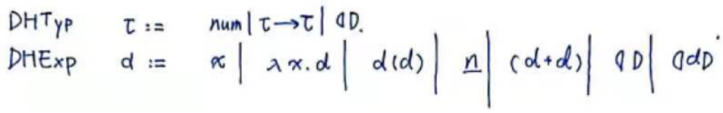

Fig. 2. Syntax

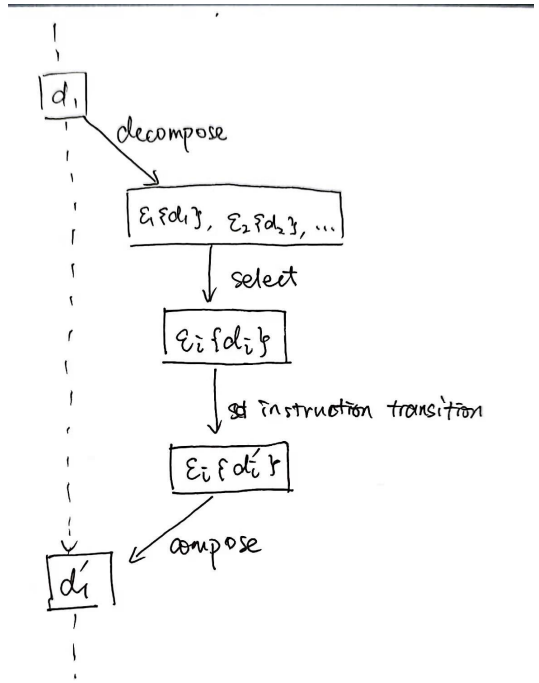

Fig. 3. Sturcture of Stepper

The structure of the stepper is shown in Figure 3. The expression is first decomposed into many independent evaluation contexts. Then, user may choose one of them to progress. And the stepper will compose them into the result then. For example, in Figure 1, we have 3 evaluation contexts highlighted as green boxes. And when we click the second one, only the expression in second box is evaluated and composed into the whole expression.

Also, for some unnecessary steps, the stepper can detect it and then pause unless user click it manually. So, during the decompose, we need to detect this situation so that the stepper will not go on again. An example is shown in Figure [].....

We now formally define the instruction transition judgement. We use $d_1 \longrightarrow d_2$ as instruction transition judgement. Figure 4 shows some of the transition judgement.

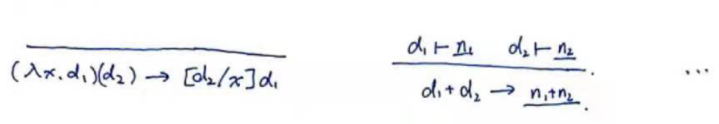

Fig. 4. Instruction Transition

In Hazel, we have four types of expressions: boxed value, indeterminate, step, and paused. The instruction transition can be implemented to provide the type of the expressions. Therefore, we can easily know whether an expression is final, paused or steppable.

Then, we have the evaluation context. Unlike the regular evaluation context, the decompose function will return a list of contexts. The recursive definition of the decompose function is shown in Figure 5.

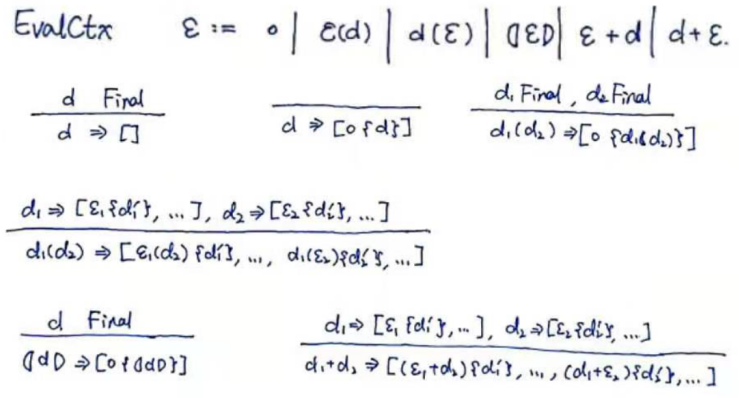

Fig. 5. Decompose

In conclusion, we first decompose the expression into many evaluation contexts for user to choose. Then, we run the instruction transition and compose the result to user.

ACKNOWLEDGMENTS

To Robert, for the bagels and explaining CMYK and color spaces.