Arithmetische Ausdrücke in Rust Dokumentation

Mario Occhinegro (74661)
Michael Kirmizakis (75592)
Jonas Heck (67889)

Inhaltsverzeichnis

1 Datatype

Einfaches Enum zur Bestimmung der Art des Datentyps. Wir haben Integer oder boolsche Werte

2 Expression Enum

Das Expression Enum beschreibt alle möglichen Ausdrücke, die wir verarbeiten können. Eine Variante jeweils für die Zahlen von 1-9. Sowie jeweils eine Variante für true und false. Für die Operatoren Plus, Mult, Or und And gibt auch jeweils eine Variante. Interessant ist hier, dass diese hier wieder Unterexpressionen verwalten. Ein mal für Rechts und ein mal für Links.

3 Evaluieren von Expressions

3.1 Verwendung des Pattern-Matching zur Evaluation von Ausdrücken

Um eine Expression zu evaluieren, wird das Pattern-Matching von Rust verwendet. Bezüglich des Rückgabetyps, ist zu beachten, dass aufgrund der Tatsache, dass in den Ausdrücken sowohl logische Ausdrücke als auch Ausdrücke mit Zahlen vorkommen, hier eine Unterscheidung benötigt wird. Weiterhin besteht die Möglichkeit, dass ein Ausdruck gar nicht evaluiert werden kann, weil eventuell eine nicht auflösbare Vermischung von logischen und mathematischen Ausdrücken vorkommt. Manche dieser vermischten Ausdrücke können jedoch aufgrund der Short-Circuit-Evaluation trotzdem ausgewertet werden.

3.2 Rückgabetyp und Unterscheidung von Zahlen und logischen Werten

Um nun den Rückgabewert der Evaluate-Funktion zu definieren, wird
Option <Result <i32, bool >>gewählt, da mittels der Option, angegeben werden kann, ob überhaupt ein Wert zurückgegeben wird (Some), oder ob die Auswertung gescheitert ist und kein Wert zurückgegeben werden kann (None). Um zwischen einer Zahl und einem logischen Wert zu unterscheiden, wird
Result <i32, bool >verwendet, hierdurch, kann mittels Ok(number) eine Zahl und mittels Err(logic) ein logischer Wert zurückgegeben werden, dies wird für das Pattern-Matching benötigt, da es hierdurch einfacher wird, den Typ einer Evaluation zu matchen und zu erkennen.

3.3 Evaluation von verschiedenen Ausdruckstypen

3.3.1 Evaluation von Zahl-Expressions

Um die eigentliche Evaluation durchzuführen, werden zunächst die Patterns für Expression angegeben, die nur eine Zahl enthalten. Das bedeutet, es werden die Expression-Patterns Zero bis Nine im Pattern-Matching angegeben, und die entsprechende Zahl wird zurückgegeben. Ebenso werden die Patterns für die booleschen Werte true und false definiert.

3.3.2 Evaluation von Plus-Expressions

Als nächstes folgt das Pattern für eine Plus-Expression. Hier werden zunächst beide Seiten der Plus-Expression evaluiert. Anschließend wird ein Pattern-Matching auf die beiden Ergebnisse angewendet. Wenn beide Ergebnisse Zahlenwerte liefern (Some(Ok(left_value))), werden die beiden Werte von der linken und rechten Seite aufaddiert und an die aufrufende Funktion zurückgegeben. Wenn eine der beiden Seiten keine Zahl zurückliefert, wird dies mittels des don't care Patterns _ im entsprechenden Fall erkannt, und es wird None zurückgegeben, um anzuzeigen, dass keine Auswertung der Expression möglich war.

3.3.3 Evaluation von Multiplikation-Expressions

Als nächstes erfolgt die Pattern-Prüfung für eine Multiplikations-Expression. Hier wird zuerst überprüft, ob an der linken Stelle der Expression eine 0 steht. In diesem Fall kann die weitere Evaluierung übersprungen werden, und direkt 0 zurückgegeben werden , da 0 * _ immer 0 ergibt. In allen anderen Fällen wird die linke Seite der Expression evaluiert. Anschließend wird ein Pattern-Matching auf das Ergebnis dieser Evaluation angewendet. Falls die Evaluierung der linken Seite der Expression ergibt, dass diese 0 ist, wird direkt 0 zurückgegeben. Wenn die linke Seite zu einer Zahl ungleich 0 evaluiert wurde, wird auch die rechte Seite evaluiert. Falls die linke Seite gar nicht evaluiert werden konnte, wird mittels des don't care Patterns None zurückgegeben. Nachdem die rechte Seite evaluiert wurde, wird auch auf diesem Ergebnis ein Pattern-Matching durchgeführt, das dieselben Fälle wie das Pattern-Matching für die linke Seite abdeckt. Wenn ein Zahlenwert ungleich 0 ermittelt wurde, wird anders als im Fall für die linke Seite, die Zahlen der linken und rechten Seite multipliziert und zurückgegeben.

3.3.4 Evaluation von Oder-Expressions

Um eine Oder-Expression auszuwerten, wird zunächst die linke Seite ausgewertet, und dann wird ein Pattern-Matching auf diesem Ergebnis durchgeführt. Falls die linke Seite nicht ausgewertet werden konnte, wird direkt None zurückgegeben. Wenn im Rahmen der Auswertung der linken Seite eine Zahl ermittelt wurde, wird die Auswertung ebenfalls abgebrochen und None zurückgegeben. Wenn die linke Seite jedoch zu einem logischen Wert ausgewertet werden konnte, wird auf diesem Wert wiederum ein Pattern-Matching gestartet. Wenn die linke Seite der Oder-Expression zu true ausgewertet wurde, kann direkt true als Wert für die Expression zurückgegeben werden, da die rechte Seite das Ergebnis der Auswertung nicht mehr verändern kann. Wenn die linke Seite zu false ausgewertet wurde, wird auch die rechte Seite ausgewertet, und dann wird auf diesem Ergebnis ebenfalls ein Pattern-Matching durchgeführt. Dabei wird geprüft, ob die Expression überhaupt evaluiert werden konnte und ob ein boolescher Wert für die rechte Seite evaluiert wurde. Falls ein boolescher Wert auf der rechten Seite steht, wird dieser als Wert der Expression zurückgegeben, da der Wert der Expression nun nur noch von diesem abhängt.

3.3.5 Evaluation von Und-Expressions

Abschließend folgt das Pattern für die Und-Expression. Bei dieser wird ebenfalls zuerst die linke Seite der Expression ausgewertet, und auf dem Ergebnis dieser Auswertung wird ein Pattern-Matching durchgeführt. Falls die linke Seite nicht evaluiert werden konnte, wird None zurückgegeben. Wenn die linke Seite zu einer Zahl evaluiert wurde, wird die weitere Auswertung abgebrochen und None zurückgegeben. Nur wenn die linke Seite zu einem booleschen Wert ausgewertet wurde, wird auf diesem Ergebnis ein weiteres Pattern-Matching durchgeführt. Dabei wird zunächst geprüft, ob der ermittelte Wert false ist. In diesem Fall wird die rechte Seite gar nicht ausgewertet, da bei einer Und-Expression sowohl die rechte als auch die linke Seite true sein müssen, um true als Ergebnis zurückzugeben. Wenn der linke Teil der Und-Expression true ist, wird auch der rechte Teil ausgewertet und geprüft, ob die Auswertung einen Wahrheitswert ergibt. Wenn die Auswertung der rechten Seite einen booleschen Wert ergibt, wird dieser Wert als Ergebnis der Expression zurückgegeben, da der Wert der Expression nun nur noch von diesem abhängt.

4 Parsen von Expressions

Die Funktion, die das Parsen von Expression zuständig ist, funktioniert wie folgt. Sie bekommt einen String und ein Präzedenzlevel, um sich in der Rekursion zu merken was die Präzedenz der Parent-Expression ist. Die Ausgabe ist dann ein Expression-Enum, das die eingegebene Expression beschreibt. Innerhalb des Rumpfes bewegen wir uns von links nach rechts durch den übergebenen String. Jeder Aufruf fängt damit an, dass wir den Charakter auf dem wir uns befinden parsen. Dies ist immer eine Zahl, ein Boolean oder ein geklammerter Ausdruck. Diese Expression merken wir uns als linken Teilbaum. Danach betrachten wir, falls vorhanden, den nächsten Charakter, bei dem es sich in unserer Logik um einen Operator handeln muss, da wir keine unnären Operatoren unterstützen. Falls nicht vorhanden, geben wir die linke Seite als Expression zurück und sind fertig. Für den Fall, dass es sich aber wirklich um einen Operator handelt, bestimmen wir zu aller erst dessen Präzedenz. Falls die übergeordnete Präzedenz der Parent Expression höher ist, so brechen wir an dieser Stelle ab und geben den bis dato errechnetem Baum zurück. Das ist notwendig, da falls die Präzedenz des übergeordneten Operators größer ist, der momentane Charakter teil der Parent-Expression ist. Falls für den anderen Fall setzen wir den Operator in die Wurzel unseres Baums. Jetzt fehlt nur noch die rechte Seite unseres Baumes. Diese berechnen wir rekursiv auf dem Rest des Iterables und dem momentanen Präzedenz-Operator. Falls der Unteraufruf den String nicht voll aufbraucht, springen wir wieder in die While-Schleife der höheren Rekursionsebene. Jetzt updaten wir den linken Teilbaum mit dem berechneten Ergebnis. Dies tun wir so lange, bis keine Characters mehr übrig sind.

5 Expression Typcheck

Diese Methode prüft den Datentyp eines Ausdrucks und gibt ein Ergebnis zurück. Wenn der ausdruck eine Zahl zwischen 0 und 9 ist, wird der Datentyp TInt zurückgegeben. Wenn der Ausdruck ETrue oder EFalse ist, wird der Datentyp TBool zurückgegeben. Wenn der Ausdruck eine Addition, Multiplikation, Logische-Oder oder logische Und-Operation ist, werden die Datentypen der linken und rechten Operanden überprüft. Wenn beide Operanden den richtigen Typ haben, wird TInt bzw. TBool zurückgegeben, andernfalls wird None zurückgegeben.