—- Modelle—-

Datenarten

Michael Felke Martina Mostert

Revision 2010-03-23

In Zusammenarbeit mit

Marc Horst, Christian Janßen

Lizenz

Verwendung des Modells

Der Inhalt dieses Dokuments führt ein owl's-Modell gemäss den nachstehenden Richtlinien, Konditionen und Hinweisen aus. Dieses Dokument stellt keine Verpflichtung dar irgendeinen Teil dieser Spezifikationen in irgendein Produkt einer Firma zu implementieren. An den in diesem Dokument enthaltenen Informationen sind Änderungen vorbehalten.

Lizenzbedingungen

In Abhängigkeit mit allen nachstehenden Richtlinien und Bedingungen gewährt Ihnen das owl's-Team als Inhaber der Urheberrechte an diesem owls's-Modell hiermit eine unentgeltliche, nicht exklusive, nicht übertragbare, unbefristete, weltweite Lizenz (ohne das Recht Unterlizenzen zu erteilen), dieses owl's-Modell zu nutzen, um Software und spezialisierte Modelle zu erstellen und vertreiben, die auf diesem owl's-Modell basieren, sowie dieses owl's-Modell zu nutzen, kopieren und zu verteilen gemäss dem Urheberrechtsgesetz; vorrausgesetzt dass:
(1) beide, der Urheberrechtshinweis wie oben kenntlich gemacht und dieser Lizenztext auf allen Kopien dieses owl's-Modells erscheinen;
(2) dieses owl's-Modell Informationszwecken dient und nicht zu kommerziellen Zwecken über einen Netzwerk-Rechner kopiert oder verbreitet oder über irgendein anderes Medium gesendet oder anderweitig weiterveräussert oder weitergegeben wird; und
(3) keine Änderungen an diesem owl's-Modell vorgenommen werden.
Diese eingeschränkte Lizensierung endet fristlos, falls gegen irgendeine dieser Richtlinien oder Bedingungen verstossen wird. Nach Beendigung sind unverzüglich alle im Besitz befindlichen Kopien dieses owl's-Modells zu vernichten.

Patente

Das Augenmerk von Anwendern wird auf die Möglichkeit gelenkt, dass die Einhaltung oder Übernahme von owl's-Modellen die Verwendung von urheberrechtlich geschützten Erfindungen erfordern kann. Das owl's-Team ist nicht verantwortlich dafür, Patente zu kennzeichnen, für die bei Beachtung eines owl's-Modells möglicherweise die Pflicht zur Lizensierung besteht, oder die Einziehung von Erkundigungen über die Rechtsgültigkeit oder den Geltungsbereich von solchen Patenten, die ihm zur Kenntnis gebracht wurden, durchzuführen. owl's-Modelle stellen lediglich eine Richtline bzw. technische Norm dar. Interessierte Anwender sind selbst dafür verantwortlich sich gegen Haftung bei Patentverletzungen zu schützen.

Allgemeine Verwendungseinschränkungen

Jede unberechtigte Verwendung dieses owl's-Modells kann Urheberrechte, Markenrechte, Bestimmungen und Satzungen des Nachrichtenwesens verletzen. Dieses Dokument enthält Informationen die urheberrechtlich geschützt sind. Alle Rechte vorbehalten. Kein in diesem Werk enthaltener, durch das Urheberrecht geschützter Teil darf nachgemacht oder in irgendeiner Form oder auf irgendeine Art verwendet werden - grafisch, elektronisch oder mechanisch, einschlieslich fotokopieren, aufzeichenen, aufnehmen oder Informationsaufzeichnungssysteme und Informationswiedergabesystemen - ohne die Zustimmung des Urheberrechte Inhabers.

Haftungsausschluss

Obwohl davon ausgegangen wird, dass diese Veröffentlichung fehlerfrei ist, wird sie ohne Mängelgewähr zur Verfügung gestellt und kann daher Fehler oder Druckfehler enthalten. Das owl's-Team gewährt keinerlei Garantie in irgendeiner Form, ausdrücklich oder stillschweigend, bezüglich dieser Veröffentlichung, einschlieslich, aber nicht beschränkt auf, jegliche Gewährleistung wegen Rechtsmängel oder Eigentum, einschliesslich Zusicherung allgemeiner Gebrauchstauglichkeit oder Gewährleistung der Eignung für bestimmte Zwecke oder Nutzung. In keinem Fall ist das owl's-Team haftbar zu machen für hierin enthaltene Fehler oder für direkte, indirekte, zufällige, konkrete, Folge-, Vertrauens- oder Sachschäden, einschliesslich entgangener Gewinne, Einnahmen, Daten oder Nutzen, zugezogen durch jeden Benuzter oder Dritte in Zusammenhang mit der Einrichtung, Durchführung oder Nutzung dieses Materials, selbst wenn auf die Möglichkeit solcher Schäden hingewiesen wurde. Das gesamte Risiko bezüglich Qualität und Leistungsfähigkeit der mit dieser Spezifikation entwickelten Software wird von Ihnen getragen. Dieser Haftungsausschluss stellt einen wesentlichen Teil der gewährten Lizenz zur Nutzung dieses owl's-Modells dar.

Markenzeichen

Alle Produkt- und Firmennamen dienen lediglich Bezeichnungs-Zwecken und können Markenzeichen ihrer jeweiligen Inhaber sein.

Modellkonformität

Das owl's-Team legt fest, dass es die einzige Instanz ist und sein wird, die Entwickler, Lieferanten und Verkäufer von Computer Software dazu berechtigen kann, Zertifizierungsmarken, Markenzeichen oder andere spezielle Bezeichnungen zu verwenden, um die Übereinstimmung mit diesen Materialien zu kennzeichnen. Für Software, die in Übereinstimmung mit dieser Lizenz entwickelt wurde, kann dann und nur dann Konformität mit diesem owl's-Modell erklärt werden, wenn die Konformität der Software darin besteht, dass sie vollständig in allen Punkten mit den Konformitätsanforderungen übereinstimmt, so wie diese in dem owl's-Modell festgelegt sind. Für Software, die nur teilweise die angegebenen Konformitätsanforderungen einhält, kann lediglich geltend gemacht

werden, dass die Software auf diesen owl's-Modellen basiert, kann jedoch nicht die Konformität mit diesen owl's-Modellen erklären. Im Falle, dass Test-Suits für dieses owl's-Modell vom owl's-Team implementiert oder anerkannt werden, kann für eine Software, die dieses owl's-Modell benutzt, nur dann die Konformität mit diesen owl's-Modellen erklären, wenn die Software die Test-Suits einwandfrei durchläuft.

Problemmeldung

Alle owl's-Modelle werden fortlaufend überprüft und verbessert. Als Teil dieses Prozesses möchten wir die Leser dazu auffordern, evtl. gefundene Zweideutigkeiten, Widersprüche und Ungenauigkeiten an die Adresse mail@owl-s.org zu schicken.

Inhaltsverzeichnis

1 Was sind Datenarten

Datenarten dienen dazu Daten unabhängig von der verwendeten Technik konsistent übertragen und speichern aber trozdem flexibel verarbeiten zu können. Hierzu werden die Daten anhand der Eigenschaften ihres jeweiligen Wertebereichs abstrahiert. Die vielfältigen Möglichkeiten, Daten zu be- und verrechnen, bleiben dabei jedoch unberücksichtigt.

1.1 Daten, Wertebereich und Mengen

Daten sind, präziser formuliert, die Abbildung von Elementen einer oder mehrerer Mengen auf definierte – meist elektronische – Zustände, die innerhalb eines Computers verarbeitet werden können. Die Gesamtheit der abgebildeten Elemente wird Wertebereich genannt. Da diese Abbildung, abhängig von der verwendeten Technik, auf unterschiedliche Weise durchgeführt werden kann, ist es notwendig die Daten unabhängig von dieser zu betrachten und lediglich die durch die Daten repräsentierten Elemente der Mengen zu betrachten. Somit sind Daten von der gleichen Art, wenn sie Elemente derselben Mengen repräsentieren, sprich, denselben Wertebereich besitzen.

Daten können mehrere Erscheinungsformen haben z.B. Binärzahlen $< Bitfolge; Zahlenwert >$, Zeichencode $< Codenummer; Zeichen >$ oder C-Unions. So können auch Elemente verschiedener Mengen die gleiche Repräsentation besitzen. Erst die Verwendung der Daten entscheidet über die betrachtete Menge. Wenn eine Datenart mehrere Mengen repräsentiert, muss eine der Mengen den gesamten Wertebereich abdecken, [1] so dass der gesamte Wertebereich der Datenart, dem der grössten Menge entspricht. Die Mengen müssen aber nicht auch noch gleich mächtig sein.

1.2 Datenarten sind keine Datentypen

Die Datenart ist nicht gleichbedeutend mit dem Typ der Daten. Ein Datentyp definiert gültige Operationen auf Daten, evtl. unterschiedlicher Art, welche aus gegebenen Werten andere Werte ableiten. Der Wertebereich eines Datentyps ist häufig nicht klar definiert und leitet sich meist aus den Operationen ab. Datenarten hingegen definieren die von den Daten abgebildeten Mengen und Methoden zum Speichern, Übertragen und Prüfen der Daten. Die Datenarten dienen im Rahmen von Owl's als technikunabhängige Basis zur Definition von Datentypen. Es kann daher mehrere Datentypen geben, die ein und dieselbe Datenart benutzen. Um die Daten ablegen und übertragen zu können gibt es meist mehrere, von der Technik abhängende, Formate. Welches Format später Anwendung findet, ist für die Definition von Datenarten unerheblich. So kann ein und dieselbe

[1] bei C-Unions und äquivalenten Konstrukten handelt es sich nicht um die Repräsentation mehrere Mengen sondern um eine Vereinigungsmenge

Art von Daten in unterschiedlichen Formaten abgelegt werden, ohne ihre Anwendung zu beeinflussen[2]. Die Operationen von Datentypen werden jedoch für bestimme Kodierung(en) der Werte entwickelt. Das für die Daten eines Typs verwendete Format kann daher nicht gewechselt werden, ohne die Leistungsfähigkeit der Anwendung - positiv oder negativ - zu beinflussen. Die Definition von Datentypen mittels Datenarten ermöglicht es nun zum einen Programme unabhänig vom Format zu schreiben und andererseites das optimale Format für Verarbeitung eines Programms auf einem Rechner zuverwenden.

1.3 Datenartenverbände

Verbände von Datenarten zur strukturierten Ablage von Daten (struct, union, array, etc) sind ebenfalls wieder Datenarten.

1.4 Hierarchiebäume von Datenarten

Ähnlich wie Datentypen lassen sich auch Datenarten durch Ableitungsbeziehungen ordnen. Die so erstellten Hierarchien orientieren sich allerdings nur an den Eigenschaften der Datenarten bezüglich ihrer Darstellung, ihrer Wertebereiche und der repräsentierten Mengen, aber nicht an der Anwendung bzw. verfügbaren Operationen. Da es keine allgemeine Wurzeldatenarten gibt, von der sich alle anderen ableiten, handelt es sich jedoch nur um partielle Hierachisierung zu bestimmten Themenkreisen. Dabei kann es vorkommen, dass eine Datenart in verschiedenen Hierarchien an unterschiedlicher Position auftaucht.

1.5 Metadaten

Metadaten sind Daten über Daten, zu diesen gehören:

- Beschreibungsdaten sind Metadaten, welche dazu dienen eine Datenart bzw. ihre Instanz zu beschreiben. Sie sind abhängig von Definition, Speicherort oder Wert der Daten auf die sie sich beziehen. Hierzu zählen unter anderem Angaben zu Grösse, Definition, Parametrisierung, Struktur und Wertebereich der Daten. Ihr genauer Umfang hängt von der Datenart ab.
- Verwaltungsdaten sind Metadaten, welche zur Verwaltung der Daten benötigt oder im Rahmen der Verwaltung erfasst werden können. Sie sind unabhängig von den eigentlichen Daten. Zu ihnen zählen Angaben zu Berechtigungen, Erstellungzeitpunkten, Zugriffsstatistiken etc. Ob und in welchen Umfang Verwaltungsdaten verfügbar sind, hängt vom System ab.

[2] bzgl. Speichern, Übertragen und Prüfen

2 Eigenschaften

2.1 Definition

implizit Die Definition impliziter Datenarten ist durch Spezpacks festgelegt und damit bekannt.

explizit Die Definition expliziter Datenarten wird mit den Daten mitgeliefert und ist nicht in Spezpacks enthalten. Sie benötigen andere explizit oder implizit definierte Datenarten.

Datenarten können aufgrund ihrer Entwicklung sowohl explizit als auch implizit sein.

2.2 Verwendbarkeit

einfach Mit einer einfachen Datenart können nur Daten deklariert, aber keine Datenarten definiert werden.

erweitert Eine erweiterte Datenart kann entweder zur Definition neuer Datenarten dienen oder direkt zur Deklaration von Daten verwendet werden.

template Mit einer Template-Datenart können nur neue Datenarten definiert werden. Sie nehmen im Minimalumfang maximal eine Datenart-Id als Parameter entgegen.

2.3 Art des Wertebereichs

fest Der Wertebereich hat für alle Daten dieser Art immer den gleichen Umfang.

abhängig Der Wertebereich eines Datums ist abhängig von technischen Eigenschaften des verwendeten Computeraggregats.

begrenzbar Der Wertebereich wird durch Informationen innerhalb des Datums auf einen Teil des eigentlich möglichen Wertebereichs begrenzt.

veränderlich Der Wertebereich ändert sich im Laufe der Existenz des Datums

unendlich Der Wertebereich ist theoretisch unendlich groß

2.4 Struktur

Einteilung der Datenarten anhand der Struktur ihrer Wertebereiche.

atomar sind Datenarten mit einem definierten Wertebereich, der nicht auf andere Datenarten rückführbar ist. Atomare Datenarten müssen immer implizit definiert sein, d.h. sie können nur über Standarddefinitionslisten definiert werden. Die Elemente der Mengen von atomaren Datenarten sind nicht natürlichen Zahlen, beginnend mit Null, zugeordnet.

molekular sind Datenarten mit einem definierten Wertebereich, der sich auf die Kombination anderer Datenarten zurückführen lässt, aber nur als Ganzes behandelt werden. Molekulare Datenarten können mit Hilfe anderer molekularer oder atomarer Datenarten explizit definiert werden. Da die repräsentierten Mengen dem Rechner für die Verarbeitung der Datenart nicht bekannt sein müssen, brauchen diese bei der expliziten Definition nicht angegeben werden. Bei der Bearbeitung von Werten der Datenart muss die entsprechende Menge aber implizit bekannt sein.

modular Der Wertebereich einer modularen Datenart wird aus anderen Datenarten zusammengesetzt. Die einzelnen Teildaten lassen sich einzeln ändern.

vereinigt Vereinigung aus anderen Datenarten, ohne doppelte. Dies entspricht der üblichen Vereinigungsmenge [1]

multipel Die Zuordnung kann ggf. explizit angeben werden.

2.5 Speicherplatzbedarf

Eine wichtige Einteilung der Datenarten ist anhand ihres Speicherplatzbedarfs. Hierbei ist nicht die eigentliche Größe des belegten Speichers von Interesse, sondern inwieweit sich dieser anhand der Datenart bzw. der Daten bestimmen lässt.

statisch Der Speicherplatzbedarf liegt mit ihrer Definition der Datenart fest. Alle Daten dieser Art belegen daher immer gleichviel Speicher.

individuell Der Speicherplatzbedarf hängt von im Datum hinterlegten konstanten Informationen ab. Ein Datum beansprucht soviel Speicher, wie beim Anlegen durch Initialisierung festgelegt wurde.

dynamisch Der Speicherplatzbedarf richtet sich nach dem aktuellen Wert. Das Speichern neuer Werte kann daher eine Änderung des Platzbedarfs und damit des zugeordneten dynamischen Speicherbereichs nach sich ziehen.

[1] Teilmengen werden nicht berüchtigt, da sie keine neuen sondern nur den reduzierten Wertebereich von bereits bestehenden Datenarten erfordern.

offen Ein Beispiel für eine Datenart mit offenem Speicherplatzbedarf ist ein nativer C-String. Der Speicherplatzbedarf entspricht den gerade gespeicherten Daten. Es gibt keinen zugeordneten dynamischen Speicherbereich. Änderungen können daher zum Überschreiben anderer Daten führen.

3 Darstellung von Datenarten

Die Darstellung von Datenarten in bzw. die Abbildung auf Speicherzellen ist nur innerhalb einer Maschine, aber nicht innerhalb eines Computeraggregats gleich. Eine ggf. nötige Umwandlung der Darstellung zum Austausch der Datenarten innerhalb eines Computeraggregats wird von den jeweiligen E/A-Ports wahrgenommen. Der Wertebereich aller nicht nur maschinenweit gültigen Datenarten ist jedoch innerhalb des Computeraggregats gleich. Die transistente Datenart-Identifikation ist innerhalb eines Computeraggregats eindeutig. Datenarten haben allerdings keine Defaultwerte. Der Wert eines neu angelegten Datums ist undefiniert, sofern es nicht explizit mit einem Wert initialisiert wurde. Genauer gesagt richtet sich der Wert des Datums nach dem Inhalt des Speichers, welcher vom Computeraggregat für das Datum verwendet wird. Da sowohl die Behandlung des Speichers durch das Computeraggregat als auch die Darstellung des Wertes im Speicher unbekannt sind, dürfen keinerlei Annahmen über eventuelle Werte von nicht initialisierten Daten gemacht werden.

3.1 Metadaten

Die Parameter einer Dateninstanz bzw. eines Wertes, welche neben der Datenart noch alle beim Angelegen der Dateninstanz (z.B. die Anzahl Elemente eines statischen Arrays) oder beim Schreiben eines Datenwerts (z.B. Anzahl Zeichen einer Zeichenkette) verwendet wurden, fließen in ihre Darstellung ein. Ausserdem kann die Darstellung einer Datenart auch Platz zur Speicherung weiterer Metadaten vorsehen. Dies schliesst auch eine eventuelle Speicherung von Daten bzw. Metadaten an einem anderen Ort als der eigentlichen Dateninstanz mit ein.

3.2 Normdarstellung

Zu jeder persistenten Datenart existiert eine Normdarstellung für den Datenaustausch zwischen verschieden Computeraggregaten. Die Normdarstellung verwendet Big-Endian und hält sich ansonsten an soweit wie möglich an internationale Standards.

Für die Normdarstellung wird eine als Byte bezeichnete ”Standardzelle” mit 256 verschiedenen Zuständen zur Speicherung von Daten zugrunde gelegt. Das heisst, jedes physikalische Speichermedium für eine Normdarstellung muss über einzeln adressierbare Zellen verfügen, welche mindestens 256 Zustände unterscheiden können. Das Byte der Normdarstellung ist aber nicht als 8Bit-Binärzahl aufzufassen. Vielmehr abstrahiert die Normdarstellung von üblichen Speicherorganisationen und erlaubt damit auch keine Zerlegung der 256 Zustände in Bits oder ähnlichem.

3.3 textuelle Dumpdarstellung

Zu jeder Datenart wird eine standardisierte Darstellung auf einen minimalen Zeichensatz, den so genannte Dumpzeichensatz, definiert. Diese so genannte Dumpdarstellung ist eindeutig und vollständig, so dass jeder mögliche Wert einer Datenart im Dump dargestellt werden kann und dass mittels der Dumpdarstellung auch jeder Wert der Datenart angegeben werden kann.

Ziel ist es, eine für Menschen - im Vergleich zur Normdarstellung - einfache les- bzw. editierbare Form der Daten zu haben, welche auf nahezu jedem System zur Verfügung gestellt werden kann. Sie richtet sich nicht nach kulturellen Gepflogenheiten sondern nur nach datentechnischen Gesichtspunkten.

Mathematisch handelt es sich um eine bijektive Abbildung der Worte über dem Dumpzeichensatz auf den Wertebereich einer Datenart.

Diese Form der textuellen Darstellung ist sehr präzise und streng. Zu jedem Wert gibt es genau eine Zeichenfolge. Was den Vergleich von Werten durch zeichenweisen Vergleich der Dumpdarstellungen ermöglicht. Darüber hinaus werden im Rahmen dieses Modells und der Datenartspezifikationen keinerlei Festlegungen über die Standard-Textdarstellungen der Werte von Datenarten für bestimmte Sprachen (z.B. Programmiersprachen, Datenbeschreibungssprachen) getroffen. Dies wird den für die Sprache Zuständigen überlassen.

Bei Datenarten, die es erlauben, die Elemente ihres Wertebereichs frei anzugeben, muss für die Elemente ein dumpkonformes Mnemonik angeben werden. Dumpkonform bedeutet, dass es sich nur um Zeichen des Minimalzeichensatzes handelt und diese obendrein keine Steuerzeichen/Leerzeichen oder die für die Strukturierung des Dumps reservierten Zeichen enthalten. Optional kann ein Dump auch ohne Mnemoniks erzeugt werden.

3.4 Identifikation

Die Gesamtheit der Datenarten wird durch ein CROWD [1] repräsentiert. Dieses CROWD wird Datenart-Id genannt. Ein Wert des veränderlichen Wertebereichs der Datenart-Id identifiziert eine Datenart. Die Werte der Instanzen der Datenart-Id sind nur innerhalb ihrer aktuellen Ablaufumgebung gültig. Sie bilden die zu identifizierenden Datenarten zwar eindeutig (surjektiv), aber nicht eineindeutig (bijektiv) ab. D.h. innerhalb der Ablaufumgebung wird durch einen Id-Wert nur eine Datenart identifiziert. Einer Datenart können aber mehrere Darstellungswerte zugeordnet sein.

3.5 dynamische Datenarten

Zur persistenten Speicherung von dynamischen Datenarten, welche weder ein eigenes Abschlussmerkmal enthalten noch begrenzt sind, wird zuerst die Größe des Datums als OFFSIZE [2] Wert gefolgt von den eigentlichen Daten abgelegt. Denn selbst dynamische Datenarten können nicht grösser sein als der grösste mögliche Speicherblock, dieser ist immer in einem OFFSIZE abbildbar.

[1] siehe ungeordnete Menge Seite 18
[2] siehe Indices, Offset und Sizes Seite 19

3.6 Definition neuer Datenarten

Die Codierung einer neuen Datenartdefinition erfolgt maschinenlesbar mit Hilfe von bereits definierten Datenarten (insbesondere von Standard-Datenarten). Zu dieser Darstellung gibt es ein Tool, das eine textuelle Dumpdarstellung erzeugen kann.

4 spezielle Daten

4.1 minimaler Zeichensatz

Zur textuellen Darstellung von Datenarten und Angabe von symbolischen Namen steht ein minimaler Zeichensatz, der sog. Dumpzeichensatz, zur Verfügung. Dieser Zeichensatz umfasst nur Zeichen, die auf nahezu allen Rechnersystemen zur Verfügung stehen. Er beinhaltet daher alle Zeichen nach 'ISO 646 - international'. Von den Steuerzeichen sind nur die Whitespace-Zeichen gemäß Standard-C[1] enthalten.

Grossbuchstaben
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Kleinbuchstaben
a b c d e f g h i j k l m n o p q r s t u v w x y z

Ziffern
0 1 2 3 4 5 6 7 8 9

Grafikzeichen
! "# % & ' ( ) * + , - . / : ; < > = ? [ ] \^_ { } | ~

Leerzeichen

␣

Steuerzeichen

Steuerzeichen		*Standard-C*
FF	(form feed)	\f
LF	(new line)	\n
CR	(carriage return)	\r
HT	(horizontal tab)	\t
VT	(vertical tab)	\v

4.2 Mnemonik

Zur Darstellung von Mnemoniks dient die Datenart LABELCHR. Als Defaultdarstellung werden jedoch Großbuchstaben verwendet. Die Länge der Mnemoniks ist auf Bytewerte/Bytebits Zeichen (Byte) begrenzt. Die Mnemoniks definieren und unterscheiden keine Werte. Nur Werte mit

[1] siehe isspace() im ANSI-C-Standard

unterschiedlicher Definition sind auch unterschiedliche Werte. Es können jedoch unterschiedliche Mnemoniks für die gleiche Definition registiert werden. Bei mehrern Mnemoniks zu einem Wert ist nicht definiert, welcher bei einem vom System erzeugten Dump verwendet wird. Die Menmoniks sind im Dump nur erläuternde Zusatzinformationen, um Eindeutigkeit über Dumps herzustellen wird der Wert im Dump auch noch durch die Nummer in der Definitionsliste repräsentiert.
Siehe auch 3.3 auf Seite 13.

4.3 Bit und Bool

Bit repräsentiert die Menge der Binärziffer 0 und 1. Es handelt sich daher um eine Untermenge der natürlichen Zahlen. Boolean repräsentiert die Menge *wahr*, $falsch$, wobei *wahr* entweder als 0 oder 1 in einem Bit dargestellt werden könnte. Um deshalb eine optimale Implementation zu ermöglichen, sind Bit und Boolean zwei verschiedene Datenarten, da sie im Grunde zwei verschiedene Mengen repräsentieren, deren gegenseitig Abbildung willkürlich gewählt werden kann.

4.4 Byte, Word, Regword

maschinenunabhängige Basisdatenarten:

- OCTED := kleinste adressierbare Einheit mit maschinenunabhängigem Wertebereich von 0 bis 255

Es gibt drei maschinenabhängige Basisdatenarten:

- BYTE := kleinste adressierbare Einheit, Teil eines WORD
- WORD := natürliches Datenwort der Maschine
- REGWORD := maximales Datenwort der Maschine, kann gerade noch am Stück gelesen werden, passt in ein Register

Es gilt: BYTE =< WORD =< REGWORD
mit n,m element aus N $size(BYTE) = 1$
$size(WORD) = n * size(BYTE)$
$size(REGWORD) = m * size(WORD) = n * m * size(BYTE)$
Die Speicherausrichtung ist immer auf WORD.

4.5 Float

[2] Float ist eine parametrisierbare Datenart zur Darstellung von binären Fliesskommazahlen. Die Parameter der Datenart geben die Wertebereiche von Mantisse und Exponent in Form der Grösse ihrer Bitfelder an. Sie sind positiv und in der Datenart BitIndex darstellbar.

[2] entspricht einem LLVM-Datentyp (siehe Anhang A.2)

4.6 Void Datenarten

[2] Es gibt drei Datenart die keine Menge abbildet und daher einen leeren Wertebereich aufweist. VOID dient im wesentlichen zur Darstellung des C/C++-Datentyps void. Seine Elemente habe die Größe 0. Die Elemente der parametrisierbare Datenart EMPTY, besitzen genau die Größe der im Parameter angegeben Datenart. Es dient im wesentlichen als Platzhalter der nicht mit kopiert bzw. geladen oder gespeichert wird. Der Speicherplatzbedarf der Datenart SPACE ist offen. Mit SPACE können nicht verwendete Speicherbereiche deklariert werden.

4.7 Zeichenvorräte

Reine Zeichenvorräte werden unabhängig von ihrer Kodierung in Datenarten abgebildet. Zur Darstellung der Zeichenvorräte können sowohl ungeordnete wie auch feste Mengen verwendet werden. Für Vorräte von fest definierten Zeichensätze wie z.B. ASCII oder ISO 859-1 (Latin 1) eignen sich feste Mengen. Um den unbegrenzten Zeichenvorrat von Unicode darzustellen eignet sich nur ein CROWD. Zeichensätze basierend auf CROWDs gehören nicht zum Minimalumfang.

4.8 Counter

Die Datenart Counter wird für allgemeine Zählschleifen und Nummerierungen verwendet. Sie hat mindestens den Wertebereich des grössten Index, der ansonsten aber Rechner-abhängig bzw. Progset-abhängig ist.

4.9 Array, Vector, Seq

[2] Zur Darstellung von sich wiederholenden Datenarten gibt es VECTOR, ARRAY und SEQ.

Bei VECTOR handelt es sich um ein molekulares Tupel gleicher, registerfähiger Datenarten. Der Zugriff auf die einzelnen Elemente erfordert spezielle Operationen, da sie nicht adressierbar sind. Art und Anzahl der enthaltenen Datenarten werden als Parameter angegeben.

Das ARRAY entspricht im Wesentlichen dem VECTOR, wobei die Elemente einzeln adressierbar sind. Es kann daher für alle Datenarten verwendet werden. Die Anzahl der Elemente ist nur durch den Wertebereich der entsprechenden Index-Datenart begrenzt.

Darüber hinaus bietet SEQ die Abbildung einer sich dynamisch ändernden Anzahl Daten der gleichen Datenart.

4.10 Struct Datenarten

[2] Ein STRUCT ist eine Aneinanderreihung der in der Parameterliste angegebenen Datenarten. Jedes Teilelement eines STRUCT ist einzeln adressierbar. Die Reihenfolge der Teilelemente im

Speicher muss aber nicht der definierten Reihenfolge der Teilelemente entsprechen. Zur Anpassung der Teilelemente an die unterschiedliche Speicherausrichtung der Datenarten können zusätzlich SPACEs eingefügt sein.

Im Gegensatz zum STRUCT sind die Teilelemente eines SLIMSTRUCT nicht im Speicher ausgerichtet und ihre Reihenfolge entspricht immer der Definition. Zusätzliche SPACEs entfallen.

4.11 Pointer

[2] Pointer können relativ oder absolut sein. Relative Pointer geben den Ort des referenzierten Gegenstandes innerhalb seines Kontextes eindeutig an, z.B. speicherrelative Pointer gelten innerhalb eines Speichersegmentes, auf das sie sich beziehen, kanalrelative können zur Adressierung innerhalb eines Kanals verwendet werden. Absolute Pointer geben den Ort des referenzierten Gegenstandes absolut an, d.h. der Pointer enthält die vollständige Zugriffsinformationen. Da evtl. mehrere Zugriffswege existieren sind diese Zeiger nicht eindeutig. Es gibt einen Universal-Pointer, in den alle anderen Pointer übertragen werden können. Dieser ist immer absolut. Universal-Pointer können sinnvoll nur in gleichartige Pointer zurück übertragen werden. Die Zurückübertragung von Universal-Pointer in spezifische Pointer ist nur dann fehlerfrei möglich, wenn vorher von diesen spezifischen Pointern in den Universal-Pointer übertragen wurde. Der Universal-Pointer muss eindeutig sein. D.h. die Übertragung von verschiedenen spezifischen Pointern in den Universal-Pointer darf nicht zu gleichen Universal-Pointern für verschiedene Objekte führen.

Neue spezifische Pointer werden eingeführt, wenn es nach Computing Modell bzw. Interfacetechnik nicht möglich ist, schon vorhandene spezifische Pointer zu verwenden. Address-Pointer verweisen auf Speicherzellen. Der GMC stellt spezielle Operationen zur Verfügung, um einen Pointer um die Grösse einer Datenart (genauer die Anzahl Speicherzellen, welche von der Datenart benötigt werden) zu inkrementieren. bzw. zu dekrementieren.

4.12 ungeordnete Mengen

Zur Repräsentation einer beliebigen ungeordneten Menge gibt es zu jeder Menge eine Datenart, genannt CROWD, deren Wertebereich Id-Werte umfasst, welche die Elemente der Menge identifizieren. Diese Datenart CROWD wird entweder explizit mittels einer Template-Datenart CROWD-SET und einem Bezeichner/Identifikation für die Menge definiert oder ist implizit aufgrund einer Standarddefinitionsliste als Standard-Identifikation vordefiniert. Die Elemente der Menge werden über Mnemoniks identifziert (siehe 3.3).

Auch die Datenarten stellen eine ungeordnete Menge dar.

CROWDS werden zur Identifikation der im Computing Modell beschriebenen Komponenten verwendet. Dabei bilden die Tasks, die Threads, die Kanäle etc. jeweils eine eigene CROWD. Die Elemente dieser CROWD sind durch eindeutige Dumpzeichenfolgen definiert, welche vom jeweiligen System abhängen und automatisch vergeben werden.

4.13 feste Mengen

Es gibt eine Template-Datenart für Mengen mit festem Umfang. Hat die Menge ein Element mehr oder weniger, so handelt es sich um eine andere feste Menge. Eine feste Menge wird definiert durch die Auflistung von Werten einer für diese Menge spezifischen Datenart. Die so definierten Mengen heissen MODICUM. Die Elemente der Menge werden über Mnemoniks identifziert (siehe 3.3).

4.14 Handels

Handels basieren auf ungeordneten Mengen mit numerischen aber nicht zwingend fortlaufend durchzählbaren Elementen.

4.15 Indices

Es wird zwischen Indices auf Daten z.B. Array- oder Sequenz-Einträge und der Angabe eines Offsets auf eine Speicheradresse unterschieden. Die Indices sind zwecks Optimierbarkeit Datenartabhängig zu definieren und unterscheiden sich daher von der Datenart Counter. D.h. mit Hilfe der Template-Datenart Index und der Datenart-ID der zu indizierenden Daten wird eine neue Datenart definiert.

4.16 Offsets und Sizes

Es gibt eine Datenart OFFSIZE mit der sowohl ein Offset auf eine Speicheradresse als auch die Grösse eines Speicherbereichs angegeben werden kann. Denn die Größe eines Speicherbereichs kann auch durch ein Offset auf sein Ende angeben werden. Ferner kann auch der Offset auf eine Speicheradresse als die Grösse des dazwischenliegenden Speicherbereichs betrachtet werden.

4.17 Integer

[2] Es werden mehrere Datenarten zur Darstellung von ganzen Zahlen spezifiziert. Die parametrisierbare Datenart DUAL dient der Darstellung von ganzen Zahlen (Integer) im Dualsystem. Der Parameter gibt die maximale Ziffernanzahl der Dualzahl (ohne Vorzeichen) an. Zur Darstellung natürlicher Zahlen dient die Datenart NATNUM.

4.18 natürliche Zahlen

Eine der am häufigsten benötigten Mengen in der Datenverarbeitung sind Teilmengen der natürlichen Zahlen einschließlich Null. Daher gibt es auch mehrere Datenarten, die zwar alle natürliche Zahlen abbilden, aber jeweils einen unterschiedlich definierten Teil davon.

Für die Speicherung von natürlichen Zahlen beliebiger Größe gibt es zwei Möglichkeiten.

1. unendlicher Wertebereich, d.h. die Daten werden mit dynamischer Grösse gespeichert und somit ist die grösste mögliche natürliche Zahl nur durch den verfügbaren Speicher begrenzt. Daraus folgt, dass diese Datenart nur speicher- aber nicht registerfähig ist. Ferner kann problemlos der Wert für keine Zahl bzw. einen leeren Eintrag gespeichert werden.

$$\text{Wertebereich} = \{NaN; 0; 1; 2; 3; \ldots\}$$

2. NATNUM mit skalierbarem Wertebereich, der Wertebereich beginnt bei Null. Der Maximalwert berechnet sich nach der Formel $2^(8*n)-1$, wobei n als Parameter von NATNUM in Form eines OCTED angegeben wird.

$$\text{Wertebereich} = \{0; 1; 2; 3; \ldots; 256^n-3; 256^n-2; 256^n-1\}$$

5 Verwaltung von ungeordneten Mengen

Zur Verwaltung von ungeordneten Menge gibt es eine zentrale Schnittstelle, mit folgendem Funktionsumfang:

- neue Mengen einrichten - unter Angabe einer Datenart für die Definition der Elemente und einer eindeutigen Kennung für die Menge.
- Mengen auflösen (bzw. freigeben)
- Elemente einer Menge hinzufügen/definieren
- Elemente einer Menge entfernen (bzw. freigeben)
- Definitionslisten hinzufügen
- Definitionslisteninhalte entfernen (bzw. freigeben)
- Prüfen, ob eine Element für die Menge definiert ist
- Definition eines Elements auslesen
- Iteration durch die Elemente der Menge
- Normdarstellung aller definierten Elemente generieren

5.1 transistente Darstellung

Die Darstellung der Id-Werte eines CROWD im Computeraggregat ist beliebig aber eindeutig. Dazu wird jedes verwendete bzw. neu definierte Mengen-Elemente registriert. Wird eine Element nicht mehr benötigt, so ist seine Registrierung zurückzuziehen. Im Zuge der Registrierung wird ein Id-Wert zugeteilt. Wird eine Element mehrfach registriert, wird der gleiche Wert zugeteilt. Der Id-Wert ist mindestens solange gültig, bis das zugehörige Element wieder aus der Registrierung entfernt wird. Dies kann nicht geschehen, bevor nicht alle Registrierungen dieses Elements und aller mit ihm definierten Elemente auch anderer ungeordneter Mengen zurückgezogen sind.

5.2 persistente Darstellung

Zur dauerhaften Identifikation von Elementen eines CROWDs unabhängig von einem Computeraggregat werden Definitionslisten verwendet.[1] Diese Definitionslisten sind in jeder Speichereinheit, in der Id-Werte persistent abgelegt werden, zu speichern. Die Identifikation der Elemente

[1] siehe auch Distribution der Datenarten Seite 23

erfolgt dann durch die Nummer der Definitionsliste in Kombination mit der Position, an welcher das Element in der Liste geführt bzw. definiert wird. Die Listennummer wird als Counter gespeichert, die Listenposition als ein Byte. Id-Werte persistenter Mengen-Elemente bestehen also aus Listennummer+Listenposition. Die Listennummer, welche im Kopf der Definitionsliste eingetragen ist, muss zwar innerhalb der Speichereinheit eindeutig aber nicht fortlaufend sein.

5.3 Standarddefinitionslisten für Datenarten

Speziell für Datenart stehen neben Listen zur expliziten Definition von Datenarten auch vordefinierte Listen mit Standard-Datenarten zur Verfügung. Die Datenarten dieser sog. Standarddefinitionslisten sind implizit definiert. Bei Standarddefinitionslisten wird im Kopf der Definitionsliste nur deren Bezeichnung und Länge eingetragen. In einer neuen Version darf die Standarddefinitionsliste nur um neue Datenarten erweitert und nicht alte Datenarten abgeändert werden. Die Version ergibt sich somit eindeutig aus der Länge der Standarddefinitionsliste. Auf explizite Angabe der Standarddefinitionsliste für den Minimalumfang der Datenarten wird verzichtet. Die Standarddefinitionsliste des Minimalumfangs hat die feste Listennummer 0. Sie ist die einzige fest vergebene Listennummer.

6 Distribution der Datenarten (Spezpacks)

Um die Datenarten zu gruppieren und zu distributieren werden ”Pakete”, sog. Spezpacks, verwendet.
Zu einem Spezpack gehören:

- formale Datenartspezifikation
- Beschreibung der Datenartspezifikation
- formale Operationsset-Spezifikation
- Beschreibung zur Operationsset-Spezifikation

Die Identifikation der Datenart ist sowohl durch die Nummer (Spezpack + Nummerierung innerhalb des Spezpacks) als auch durch den Namen der Datenart möglich.

Die Spezpacks wiederum werden in Editionen zusammengefasst, welche nicht hierachisiert sind. Ein Spezpack kann in mehreren Editionen sein.

Es gibt einen Minimalumfang an Datenarten, der auf jedem System zur Verfügung steht. Die Liste der Datenarten und deren Beschreibung ist dem entsprechenden Spezpack zu entnehmen.

6.1 Bezeichner

Bezeichner werden als Dumpzeichenfolge realisiert. Es werden nur Buchstaben, Ziffern und Bindestrich verwendet, ferner gelten folgende Einschränkungen:

- max 32 Zeichen
- nur Kleinbuchstaben
- nur a bis z , 0 - 9 und _
- keine Leerzeichen

Die Bezeichner werden zentral registriert. Dazu dienen mehre Listen - je Klasse der bezeichneten Dinge eine. Innerhalb der Klasse müssen die Bezeichner eindeutig sein. Alle Bezeichner müssen mit Spezifikation registriert werden. Private Namensräume werden nicht zugelassen.
Für Bezeichner von Standard-Datenarten dürfen nur Buchstaben und Ziffern sowie der Unterstrich verwendet werden (keine Unterscheidung zwischen Gross- und Kleinbuchstaben). Bei der Darstellung im Dump nur als Grossbuchstaben zur besseren Erkennung im Text.
System interne Benennungen werden SystemId genannt. Ihr interner Aufbau ist systemabhängig. Globale technische Benennungen werden Kennung genannt. Allgemeinsprachliche Benennungen werden Name genannt.

A Anhang

A.1 Liste der Datenarten für das Minimalsystem

Datenart	Erläuterung
VOID	Nichts (repräsentiert eine leere Menge)
DATAKIND:=CROWDSET (d=DATADEF,n=?)	zur Angabe einer Datenart
DATADEF	zur Darstellung der Definition von Datenarten
OCTED	zur Speicherbereich von 256 Zuständen (0 - 255)
BOOL	
BIT	
COUNTER	
INDEX(d:DATAKIND)	
OFFSIZE	
ARRAY(d:DATAKIND,n:INDEX(d))	
DUMPCHR	Zeichen des Dumpzeichensatzes
MEMPTR	Pointer auf Speicherbereich
UPTR	Universalpointer
WORD	Computeraggregatweit einheitliches Speicherwort
BYTE	Computeraggregatweit einheitliche Speicherzelle
REGWORD	
SEQ(d:DATAKIND)	eine sich dynamisch ändernde Anzahl Daten der Datenart d
LABELCHR:=MODICUM (d=DUMPCHR, l=DEFLIST(d)'a'-'z';'0'-'9';'-')	
LABELSEQ:=SEQ(d=LABELCHR)	Sequenz von LABELCHR
CROWDSET(d:DATAKIND, n:LABELSEQ)	Ungeordnete Menge (nur statisch)
NATNUM(n:OCTED)	natürliche Zahl mit dem Werteumfang 0 bis $2^{(8*(n+1))}-1 = 256^{(n+1)}-1$
DEFLIST(d:DATAKIND)	Liste von Werten der Datenart d, welche jeden Wert höchstens einmal enthält
MODICUM(d:DATAKIND,l:DEFLIST)	für feste Mengen

Datenart	Erläuterung
DUMPSEQ:=SEQ(d=DUMPCHR)	
PPTR	Prozedur-Pointer

A.2 Abbildung der LLVM-Datentypen auf owl's-Datenarten

LLVM	Datenart
Floating Point Types	FLOAT(m:BITIDX e:BITIDX) m und e sind die Anzahl Bits
float	FLOAT(m=23 e=8)
double	FLOAT(m=52 e=11)
fp128	FLOAT(m=112 e=15)
x86_fp80	FLOAT(m=63 e=15)
ppc_fp128	VECTOR(d=FLOAT(m=52 e=11) n=2)
Void Type	VOID
Label Type	– nach dem Compile weg –
Integer Type	DUAL(n:BITIDX)
iN	DUAL(n=N)[1]
Array Type	ARRAY(d:DATAKIND n:INDEX(DATAKIND))
[<# elements>x <el'type>]	ARRAY(d=<el'type> n=<elements>)
Vector Type	VECTOR(d:DATAKIND n:BITIDX)
«# elements> x <el'type»	VECTOR(d=<el'type> n=<elements>)
Function Type	– nach dem Compile weg –
Structure Type	STRUCT(l:SEQ(d=DATAKIND))
<type_list>	STRUCT(l=<type_list>)
Packed Structure Type	SLIMSTRUCT(l:SEQ(d=DATAKIND))
<{ <type_list>} >	PACKEDSTRUCT(<type_list>)
Pointer Type	
<type> *	MEMPTR(d=<type>)
() *	PPTR()
void *	UPTR()
Opaque Type	– nach dem compile weg –

[1] damit sind nicht alle LLVM Integer möglich.

A.3 ISO 646 - Zeichensatz

Entspricht ECMA-6.

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1-	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2-	SP	!	”	#	$	%	&	’	(	)	*	+	,	-	.	/
3-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5-	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6-	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7-	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Zeichen gehören nur zur internationalen Variante und können in den Ländervarianten andere sein.

Erläuterung Steuerzeichen

ACK acknowledge - Antwort auf ein ENQ oder Hinweis, dass Daten erfolgreich empfangen wurden

BEL bell - akkustische Systemsignal

BS backspace - Curser ein Zeichen zurück setzen

CAN cancel - Hinweis, dass die vorausgehenden Daten fehlerhaft sind bzw. nicht beachetet werden sollen

CR carriage return - Wagenrücklauf auf Zeilenanfang

DC1 device control 1

DC2 device control 2

DC3 device control 3

DC4 device control 4

DEL delete - markiert ein gelöschtes Zeichen

DLE data link escape - veranlasst, dass die folgenden Octests als Rohdaten interpretiert werden

EM end of medium - zeigt das Ende des benutzbaren Bereichs auf einem Band an

ENQ enquiry - Anfrage an das empfangende Gegenstück, ob dieses noch bereit ist

EOT end of transmission - unter Unix Anzeige des Dateiendes auf einem Terminal

ESC escape - Abbruchsignal, z.B. zum Beenden eines Menüs, Modus etc.

ETB end of transmission block - zeigt das Ende der Übertragung eines Blocks an

ETX end of text - häufig verwendet als Abbruchsignal für Programme oder Prozesse

FF form feed - Seitenvorschub

FS file seperator - zur Trennung von Datenstrukturen

GS group seperator - zur Trennung von Datenstrukturen

HT horizontal tabulator - Sprung zur nächsten Tabulatorposition

LF line feed - auf Druckern unf einigen Terminalemulationen Zeilenvorschub; unter Unix Zeilenendekennung; unter MS-DOS, Windows und einigen Netzwerk-Standards zusammen mit CR Zeilenendekennung

NAK negative acknowledge - negative Antwort auf ein ENQ oder Hinweis, dass Daten nicht erfolgreich empfangen wurden und erneut gesendet werden können

NUL null - häufig verwendet als Endezeichen für Strings

RS record seperator - zur Trennung von Datenstrukturen

SI shift in - Rückkehr zu regulärem Zeichensatz nach SO

SO shift out - Umschalten auf einen alternativen Zeichensatz

SOH start oh header - erstes Zeichen eines Nachrichtenkopfes

STX start of text - erstes Zeichen eines Nachrichtentextes

SUB substitution - wird anstelle eines Zeichens eingefügt, dass als ungültig oder fehlerhaft erkannt wurde

SYN synchronous idle - zur Synchronisation in synchronen Übertragungssystemen

US unit seperator - zur Trennung von Datenstrukturen

VT vertical tabulator - Sprung zur nächsten Zeilentabulatorposition

Index