Appendix E: Summary of Constraints

Table E.1 collects the various constraints described in the book.

TABLE E.1 Constraints

Name	Reference	Definition and Example(s)

Equality constraint	Unit 4.3	A restriction to ensure that the population of one or more predicate roles must be equal to the population of other, compatible roles.
		Example: Patient `p` has a diastolic blood pressure reading if and only if `p` has a systolic `bp` reading.
		`diastolicBPof[p] = _ ->`
		` systolicBPof[p] = _ .`
		`systolicBPof[p] = _ ->`
		` diastolicBPof[p] = _ .`

Exclusion constraint	Unit 1.7, Unit 4.3	A restriction on two or more roles (or role lists), to ensure that no tuple may instantiate more than one of those roles (or role lists) at the same time.
		Example: No person authors and reviews the same book.
		`reviews(p, b) -> !authors(p, b).`
		Example involving a join path: Nobody may review a paper authored by someone from the same institute.
		`hasAnAuthorOf(i, ppr) <-`
		` instituteOf[p] = i,`
		` authored(p, ppr).`
		`hasAReviewerOf(i, ppr) <-`
		` instituteOf[p] = i,`
		` isAssigned(p, ppr).`
		`hasAnAuthorOf(i, ppr) ->`
		` !hasAReviewerOf(i, ppr).`

Exclusive-or constraint	Unit 2.3	A restriction on two or more roles played by instances of a common type to ensure that each instance of that type plays exactly one of those roles.
		Example: Each person is male or female but not both.
		`Person(p) -> isMale(p) ; isFemale(p).`
		`isMale(p) -> !isFemale(p).`

Frequency constraint (internal or external)	Unit 4.3	A restriction on a list of one or more roles to ensure that at any given time, each instance in the population of that role list appears there a specified number of times.
		Example (internal): Each reviewer is assigned at most three papers to review.
		`positiveNrPapersAssignedTo[r] = n <-`
		` agg<<n = count()>> isAssigned(r,_).`
		`positiveNrPapersAssignedTo[_] = n ->`
		` n <= 3.`
		Example (external): No student may enroll more than twice in the same course.

TABLE E.1 (*Continued*) Constraints

Name	Reference	Definition and Example(s)

		`nrEnrollmentsFor[s, c] = n <-`
		`agg<<n = count()>>`
		`studentInvolvedIn[e] = s,`
		`courseInvolvedIn[e] = c.`
		`nrEnrollmentsFor[_, _] = n -> n <= 2.`

Inclusive-or constraint	Unit 2.3	A restriction on two or more roles played by instances of a common type to ensure that each instance of that type plays at least one of those roles.
		Example: Each valued employee is industrious or intelligent.
		`ValuedEmployee(p) ->`
		`isIndustrious(p) ; isIntelligent(p).`

Mandatory role constraint	Unit 1.5	A restriction on a single role of a predicate to ensure that each instance in the population of the role's type must play that role. *Example*: Each person was born on some date. `Person(p) -> birthdateOf[p] = _ .`

Ring constraint	Unit 2.5	One of a class of restrictions on two type-compatible arguments of a predicate. Kinds of ring constraints include irreflexivity, asymmetry, intransitivity, and acyclicity.
		Example (irreflexive): No person is a parent of himself or herself.
		`!isParentOf(p, p).`
		Example (asymmetric): If `p1` is a parent of `p2` then `p2` is not a parent of `p1`.
		`isParentOf(p1, p2) ->`
		`!isParentOf(p2, p1).`
		Example (intransitive): If event `e1` directly precedes event `e2`, and `e2` directly precedes event `e3`, then `e1` does not directly precede `e3`.
		`directlyPrecedes(e1, e2),`
		`directlyPrecedes(e2, e3) ->`
		`! directlyPrecedes(e1, e3).`
		Example (acyclic): No person is an ancestor of himself or herself, where ancestorhood is derived from parenthood.
		`isAncestorOf(p1, p2) <-`
		`isParentOf(p1, p2) ;`
		`isParentOf(p1, p3),`
		`isAncestorOf(p3, p2).`
		`!isAncestorOf(p, p).`

Continued

TABLE E.1 (*Continued*) Constraints

Name	Reference	Definition and Example(s)

Subset constraint	Unit 2.4, Unit 4.3	A restriction to ensure that the population of one or more predicate roles must be a subset of the population of other, compatible roles.
		Example: If student `s` passed course `c` then `s` was enrolled in `c`.
		`passed(s, c) -> enrolledIn(s, c).`
		Example involving a join path: If person `p` has a title `t` that applies to only one gender `g`, then person `p` must be of gender `g`.
		`personTitleOf[p] = pt,`
		`applicableGenderOf[pt] = g ->`
		`genderOf[p] = g.`

Uniqueness constraint (internal or external)	Unit 2.1 (internal), Unit 2.3 (external)	A restriction on a list of one or more roles to ensure that at any given time, each instance in the population of that role list appears there at most once. If the constrained roles come from the same predicate, the constraint is an internal uniqueness constraint; otherwise, it is an external uniqueness constraint.
		Example (internal uniqueness constraint on first role): Each person has at most one passport number. This constraint is implied by use of functional notation for the predicate declaration.
		`passportNrOf[p] = n ->`
		`Person(p), string(n).`
		Example (internal uniqueness constraint on second role): Each passport number is held by at most one person.
		`passportNrOf[p1] = n,`
		`passportNrOf[p2] = n ->`
		`p1 = p2.`
		Example (external uniqueness): Each country and state code combination applies to at most one state.
		`countryOf[s1] = c,`
		`stateCodeOf[s1] = sc,`
		`countryOf[s2] = c,`
		`stateCodeOf[s2] = sc ->`
		`s1 = s2.`

TABLE E.1 (*Continued*) Constraints

Name	Reference	Definition and Example(s)

Value constraint	Unit 1.6, Unit 4.3	A restriction on a role that specifies what values can populate that role.
		Example: The possible gender codes are "M" and "F".
		`hasGenderCode(_:gc) ->`
		`gc = "M" ; gc = "F".`
		Example: Product ratings are in the range 1 to 5.
		`productRatingOf[_] = n ->`
		`n >= 1, n <= 5.`