В данном приложении представлен программный код библиотеки AT\_KRL\_Editor.js и его интеграции в компоненты АТ-РЕШАТЕЛЯ. Текущую версию библиотеки можно найти по ссылке: <https://github.com/grigandal625/AT_KRL_Editor>.

Оглавление

[1) JSUtils.js 2](#_Toc11008571)

[2) XMLUtils.js 13](#_Toc11008572)

[3) AT\_KRL\_Editor.js 16](#_Toc11008573)

[4) AT\_KRL\_Type.js 20](#_Toc11008574)

[5) AT\_KRL\_Object.js 22](#_Toc11008575)

[6) AT\_KRL\_Expressions.js 25](#_Toc11008576)

[7) AT\_KRL\_Fact.js 43](#_Toc11008577)

[8) AT\_KRL\_Rule.js 45](#_Toc11008578)

[9) AT\_KRL\_Parser.js 50](#_Toc11008579)

[10) TKBConvertor.pas 75](#_Toc11008580)

1. JSUtils.js
2. **var** JSON
3. **if** (JSON == undefined) {
4. (**function** () {
5. **function** N(p, r) {
6. **function** q(a) {
7. **if** (q[a] !== w)
8. **return** q[a];
9. **var** c;
10. **if** ("bug-string-char-index" == a)
11. c = "a" != "a" [0];
12. **else** **if** ("json" == a)
13. c = q("json-stringify") && q("json-parse");
14. **else** {
15. **var** e;
16. **if** ("json-stringify" == a) {
17. c = r.stringify;
18. **var** b = "function" == **typeof** c && s;
19. **if** (b) {
20. (e = **function** () {
21. **return** 1
22. }).toJSON = e;
23. **try** {
24. b = "0" === c(0) && "0" === c(**new** t) && '""' == c(**new** A) && c(u) === w && c(w) === w && c() === w && "1" === c(e) && "[1]" == c([e]) && "[null]" == c([w]) && "null" == c(**null**) && "[null,null,null]" == c([w, u, **null**]) && '{"a":[1,true,false,null,"\\u0000\\b\\n\\f\\r\\t"]}' ==
25. c({
26. a: [e, !0, !1, **null**, "\x00\b\n\f\r\t"]
27. }) && "1" === c(**null**, e) && "[\n 1,\n 2\n]" == c([1, 2], **null**, 1) && '"-271821-04-20T00:00:00.000Z"' == c(**new** C(-864E13)) && '"+275760-09-13T00:00:00.000Z"' == c(**new** C(864E13)) && '"-000001-01-01T00:00:00.000Z"' == c(**new** C(-621987552E5)) && '"1969-12-31T23:59:59.999Z"' == c(**new** C(-1))
28. } **catch** (f) {
29. b = !1
30. }
31. }
32. c = b
33. }
34. **if** ("json-parse" == a) {
35. c = r.parse;
36. **if** ("function" == **typeof** c)
37. **try** {
38. **if** (0 === c("0") && !c(!1)) {
39. e = c('{"a":[1,true,false,null,"\\u0000\\b\\n\\f\\r\\t"]}');
40. **var** n = 5 == e.a.length && 1 === e.a[0];
41. **if** (n) {
42. **try** {
43. n = !c('"\t"')
44. } **catch** (d) {}
45. **if** (n)
46. **try** {
47. n =
48. 1 !== c("01")
49. } **catch** (g) {}
50. **if** (n)
51. **try** {
52. n = 1 !== c("1.")
53. } **catch** (m) {}
54. }
55. }
56. } **catch** (X) {
57. n = !1
58. }
59. c = n
60. }
61. }
62. **return** q[a] = !!c
63. }
64. p || (p = k.Object());
65. r || (r = k.Object());
66. **var** t = p.Number || k.Number,
67. A = p.String || k.String,
68. H = p.Object || k.Object,
69. C = p.Date || k.Date,
70. G = p.SyntaxError || k.SyntaxError,
71. K = p.TypeError || k.TypeError,
72. L = p.Math || k.Math,
73. I = p.JSON || k.JSON;
74. "object" == **typeof** I && I && (r.stringify = I.stringify, r.parse = I.parse);
75. **var** H = H.prototype,
76. u = H.toString,
77. v,
78. B,
79. w,
80. s = **new** C(-0xc782b5b800cec);
81. **try** {
82. s = -109252 == s.getUTCFullYear() && 0 === s.getUTCMonth() && 1 === s.getUTCDate() &&
83. 10 == s.getUTCHours() && 37 == s.getUTCMinutes() && 6 == s.getUTCSeconds() && 708 == s.getUTCMilliseconds()
84. } **catch** (Q) {}
85. **if** (!q("json")) {
86. **var** D = q("bug-string-char-index");
87. **if** (!s)
88. **var** x = L.floor,
89. M = [0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334],
90. E = **function** (a, c) {
91. **return** M[c] + 365 \* (a - 1970) + x((a - 1969 + (c = +(1 < c))) / 4) - x((a - 1901 + c) / 100) + x((a - 1601 + c) / 400)
92. };
93. (v = H.hasOwnProperty) || (v = **function** (a) {
94. **var** c = {},
95. e;
96. (c.\_\_proto\_\_ = **null**, c.\_\_proto\_\_ = {
97. toString: 1
98. }, c).toString != u ? v = **function** (a) {
99. **var** c = **this**.\_\_proto\_\_;
100. a = a **in** (**this**.\_\_proto\_\_ = **null**, **this**);
101. **this**.\_\_proto\_\_ =
102. c;
103. **return** a
104. } :
105. (e = c.constructor, v = **function** (a) {
106. **var** c = (**this**.constructor || e).prototype;
107. **return** a **in** **this** && !(a **in** c && **this**[a] === c[a])
108. });
109. c = **null**;
110. **return** v.call(**this**, a)
111. });
112. B = **function** (a, c) {
113. **var** e = 0,
114. b,
115. f,
116. n;
117. (b = **function** () {
118. **this**.valueOf = 0
119. }).prototype.valueOf = 0;
120. f = **new** b;
121. **for** (n **in** f)
122. v.call(f, n) && e++;
123. b = f = **null**;
124. e ? B = 2 == e ? **function** (a, c) {
125. **var** e = {},
126. b = "[object Function]" == u.call(a),
127. f;
128. **for** (f **in** a)
129. b && "prototype" == f || v.call(e, f) || !(e[f] = 1) || !v.call(a, f) || c(f)
130. } :
131. **function** (a, c) {
132. **var** e = "[object Function]" == u.call(a),
133. b,
134. f;
135. **for** (b **in** a)
136. e && "prototype" ==
137. b || !v.call(a, b) || (f = "constructor" === b) || c(b);
138. (f || v.call(a, b = "constructor")) && c(b)
139. } :
140. (f = "valueOf toString toLocaleString propertyIsEnumerable isPrototypeOf hasOwnProperty constructor".split(" "), B = **function** (a, c) {
141. **var** e = "[object Function]" == u.call(a),
142. b,
143. h = !e && "function" != **typeof** a.constructor && F[**typeof** a.hasOwnProperty] && a.hasOwnProperty || v;
144. **for** (b **in** a)
145. e && "prototype" == b || !h.call(a, b) || c(b);
146. **for** (e = f.length; b = f[--e]; h.call(a, b) && c(b));
147. });
148. **return** B(a, c)
149. };
150. **if** (!q("json-stringify")) {
151. **var** U = {
152. 92: "\\\\",
153. 34: '\\"',
154. 8: "\\b",
155. 12: "\\f",
156. 10: "\\n",
157. 13: "\\r",
158. 9: "\\t"
159. },
160. y = **function** (a, c) {
161. **return** ("000000" + (c || 0)).slice(-a)
162. },
163. R = **function** (a) {
164. **for** (**var** c = '"', b = 0, h = a.length, f = !D || 10 < h, n = f && (D ? a.split("") : a); b < h; b++) {
165. **var** d = a.charCodeAt(b);
166. **switch** (d) {
167. **case** 8:
168. **case** 9:
169. **case** 10:
170. **case** 12:
171. **case** 13:
172. **case** 34:
173. **case** 92:
174. c += U[d];
175. **break**;
176. **default**:
177. **if** (32 > d) {
178. c += "\\u00" + y(2, d.toString(16));
179. **break**
180. }
181. c += f ? n[b] : a.charAt(b)
182. }
183. }
184. **return** c + '"'
185. },
186. O = **function** (a, c, b, h, f, n, d) {
187. **var** g,
188. m,
189. k,
190. l,
191. p,
192. r,
193. s,
194. t,
195. q;
196. **try** {
197. g = c[a]
198. } **catch** (z) {}
199. **if** ("object" == **typeof** g && g)
200. **if** (m = u.call(g), "[object Date]" != m || v.call(g,
201. "toJSON"))
202. "function" == **typeof** g.toJSON && ("[object Number]" != m && "[object String]" != m && "[object Array]" != m || v.call(g, "toJSON")) && (g = g.toJSON(a));
203. **else** **if** (g > -1 / 0 && g < 1 / 0) {
204. **if** (E) {
205. l = x(g / 864E5);
206. **for** (m = x(l / 365.2425) + 1970 - 1; E(m + 1, 0) <= l; m++);
207. **for** (k = x((l - E(m, 0)) / 30.42); E(m, k + 1) <= l; k++);
208. l = 1 + l - E(m, k);
209. p = (g % 864E5 + 864E5) % 864E5;
210. r = x(p / 36E5) % 24;
211. s = x(p / 6E4) % 60;
212. t = x(p / 1E3) % 60;
213. p %= 1E3
214. } **else**
215. m = g.getUTCFullYear(), k = g.getUTCMonth(), l = g.getUTCDate(), r = g.getUTCHours(), s = g.getUTCMinutes(), t = g.getUTCSeconds(), p = g.getUTCMilliseconds();
216. g = (0 >= m || 1E4 <= m ? (0 > m ? "-" : "+") + y(6, 0 > m ? -m : m) : y(4, m)) + "-" + y(2, k + 1) + "-" + y(2, l) + "T" + y(2, r) + ":" + y(2, s) + ":" + y(2, t) + "." + y(3, p) + "Z"
217. } **else**
218. g = **null**;
219. b && (g = b.call(c, a, g));
220. **if** (**null** === g)
221. **return** "null";
222. m = u.call(g);
223. **if** ("[object Boolean]" == m)
224. **return** "" + g;
225. **if** ("[object Number]" == m)
226. **return** g > -1 / 0 && g < 1 / 0 ? "" + g : "null";
227. **if** ("[object String]" == m)
228. **return** R("" + g);
229. **if** ("object" == **typeof** g) {
230. **for** (a = d.length; a--;)
231. **if** (d[a] === g)
232. **throw** K();
233. d.push(g);
234. q = [];
235. c = n;
236. n += f;
237. **if** ("[object Array]" == m) {
238. k = 0;
239. **for** (a = g.length; k < a; k++)
240. m = O(k, g, b, h, f, n, d), q.push(m === w ? "null" :
241. m);
242. a = q.length ? f ? "[\n" + n + q.join(",\n" + n) + "\n" + c + "]" : "[" + q.join(",") + "]" : "[]"
243. } **else**
244. B(h || g, **function** (a) {
245. **var** c = O(a, g, b, h, f, n, d);
246. c !== w && q.push(R(a) + ":" + (f ? " " : "") + c)
247. }), a = q.length ? f ? "{\n" + n + q.join(",\n" + n) + "\n" + c + "}" : "{" + q.join(",") + "}" : "{}";
248. d.pop();
249. **return** a
250. }
251. };
252. r.stringify = **function** (a, c, b) {
253. **var** h,
254. f,
255. n,
256. d;
257. **if** (F[**typeof** c] && c)
258. **if** ("[object Function]" == (d = u.call(c)))
259. f = c;
260. **else** **if** ("[object Array]" == d) {
261. n = {};
262. **for** (**var** g = 0, k = c.length, l; g < k; l = c[g++], (d = u.call(l), "[object String]" == d || "[object Number]" == d) && (n[l] = 1));
263. }
264. **if** (b)
265. **if** ("[object Number]" ==
266. (d = u.call(b))) {
267. **if** (0 < (b -= b % 1))
268. **for** (h = "", 10 < b && (b = 10); h.length < b; h += " ");
269. } **else**
270. "[object String]" == d && (h = 10 >= b.length ? b : b.slice(0, 10));
271. **return** O("", (l = {}, l[""] = a, l), f, n, h, "", [])
272. }
273. }
274. **if** (!q("json-parse")) {
275. **var** V = A.fromCharCode,
276. W = {
277. 92: "\\",
278. 34: '"',
279. 47: "/",
280. 98: "\b",
281. 116: "\t",
282. 110: "\n",
283. 102: "\f",
284. 114: "\r"
285. },
286. b,
287. J,
288. l = **function** () {
289. b = J = **null**;
290. **throw** G();
291. },
292. z = **function** () {
293. **for** (**var** a = J, c = a.length, e, h, f, k, d; b < c;)
294. **switch** (d = a.charCodeAt(b), d) {
295. **case** 9:
296. **case** 10:
297. **case** 13:
298. **case** 32:
299. b++;
300. **break**;
301. **case** 123:
302. **case** 125:
303. **case** 91:
304. **case** 93:
305. **case** 58:
306. **case** 44:
307. **return** e =
308. D ? a.charAt(b) : a[b],
309. b++,
310. e;
311. **case** 34:
312. e = "@";
313. **for** (b++; b < c;)
314. **if** (d = a.charCodeAt(b), 32 > d)
315. l();
316. **else** **if** (92 == d)
317. **switch** (d = a.charCodeAt(++b), d) {
318. **case** 92:
319. **case** 34:
320. **case** 47:
321. **case** 98:
322. **case** 116:
323. **case** 110:
324. **case** 102:
325. **case** 114:
326. e += W[d];
327. b++;
328. **break**;
329. **case** 117:
330. h = ++b;
331. **for** (f = b + 4; b < f; b++)
332. d = a.charCodeAt(b), 48 <= d && 57 >= d || 97 <= d && 102 >= d || 65 <= d && 70 >= d || l();
333. e += V("0x" + a.slice(h, b));
334. **break**;
335. **default**:
336. l()
337. }
338. **else** {
339. **if** (34 == d)
340. **break**;
341. d = a.charCodeAt(b);
342. **for** (h = b; 32 <= d && 92 != d && 34 != d;)
343. d = a.charCodeAt(++b);
344. e += a.slice(h, b)
345. }
346. **if** (34 == a.charCodeAt(b))
347. **return** b++, e;
348. l();
349. **default**:
350. h =
351. b;
352. 45 == d && (k = !0, d = a.charCodeAt(++b));
353. **if** (48 <= d && 57 >= d) {
354. **for** (48 == d && (d = a.charCodeAt(b + 1), 48 <= d && 57 >= d) && l(); b < c && (d = a.charCodeAt(b), 48 <= d && 57 >= d); b++);
355. **if** (46 == a.charCodeAt(b)) {
356. **for** (f = ++b; f < c && (d = a.charCodeAt(f), 48 <= d && 57 >= d); f++);
357. f == b && l();
358. b = f
359. }
360. d = a.charCodeAt(b);
361. **if** (101 == d || 69 == d) {
362. d = a.charCodeAt(++b);
363. 43 != d && 45 != d || b++;
364. **for** (f = b; f < c && (d = a.charCodeAt(f), 48 <= d && 57 >= d); f++);
365. f == b && l();
366. b = f
367. }
368. **return** +a.slice(h, b)
369. }
370. k && l();
371. **if** ("true" == a.slice(b, b + 4))
372. **return** b += 4, !0;
373. **if** ("false" == a.slice(b, b + 5))
374. **return** b += 5, !1;
375. **if** ("null" == a.slice(b,
376. b + 4))
377. **return** b += 4, **null**;
378. l()
379. }
380. **return** "$"
381. },
382. P = **function** (a) {
383. **var** c,
384. b;
385. "$" == a && l();
386. **if** ("string" == **typeof** a) {
387. **if** ("@" == (D ? a.charAt(0) : a[0]))
388. **return** a.slice(1);
389. **if** ("[" == a) {
390. **for** (c = [];; b || (b = !0)) {
391. a = z();
392. **if** ("]" == a)
393. **break**;
394. b && ("," == a ? (a = z(), "]" == a && l()) : l());
395. "," == a && l();
396. c.push(P(a))
397. }
398. **return** c
399. }
400. **if** ("{" == a) {
401. **for** (c = {};; b || (b = !0)) {
402. a = z();
403. **if** ("}" == a)
404. **break**;
405. b && ("," == a ? (a = z(), "}" == a && l()) : l());
406. "," != a && "string" == **typeof** a && "@" == (D ? a.charAt(0) : a[0]) && ":" == z() || l();
407. c[a.slice(1)] = P(z())
408. }
409. **return** c
410. }
411. l()
412. }
413. **return** a
414. },
415. T = **function** (a, b, e) {
416. e = S(a, b, e);
417. e ===
418. w ? **delete** a[b] : a[b] = e
419. },
420. S = **function** (a, b, e) {
421. **var** h = a[b],
422. f;
423. **if** ("object" == **typeof** h && h)
424. **if** ("[object Array]" == u.call(h))
425. **for** (f = h.length; f--;)
426. T(h, f, e);
427. **else**
428. B(h, **function** (a) {
429. T(h, a, e)
430. });
431. **return** e.call(a, b, h)
432. };
433. r.parse = **function** (a, c) {
434. **var** e,
435. h;
436. b = 0;
437. J = "" + a;
438. e = P(z());
439. "$" != z() && l();
440. b = J = **null**;
441. **return** c && "[object Function]" == u.call(c) ? S((h = {}, h[""] = e, h), "", c) : e
442. }
443. }
444. }
445. r.runInContext = N;
446. **return** r
447. }
448. **var** K = **typeof** define === "function" && define.amd,
449. F = {
450. "function": !0,
451. object: !0
452. },
453. G = F[**typeof** exports] && exports && !exports.nodeType && exports,
454. k = F[**typeof** window] &&
455. window || **this**,
456. t = G && F[**typeof** module] && module && !module.nodeType && "object" == **typeof** global && global;
457. !t || t.global !== t && t.window !== t && t.self !== t || (k = t);
458. **if** (G && !K)
459. N(k, G);
460. **else** {
461. **var** L = k.JSON,
462. Q = k.JSON3,
463. M = !1,
464. A = N(k, k.JSON3 = {
465. noConflict: **function** () {
466. M || (M = !0, k.JSON = L, k.JSON3 = Q, L = Q = **null**);
467. **return** A
468. }
469. });
470. k.JSON = {
471. parse: A.parse,
472. stringify: A.stringify
473. }
474. }
475. K && define(**function** () {
476. **return** A
477. })
478. }).call(**this**);
479. }

482. Date.prototype.toPString = **function** () {
483. **var** day = (**this**.getDate()).toString();
484. **if** (day.length == 1) {
485. day = '0' + day;
486. }
487. **var** mh = (**this**.getMonth() + 1).toString();
488. **if** (mh.length == 1) {
489. mh = '0' + mh;
490. }
491. **var** y = **this**.getFullYear().toString();
492. **return** day + '.' + mh + '.' + y + ' ' + **this**.toLocaleTimeString();
493. }
495. **if** (!Array.prototype.indexOf) {
496. Array.prototype.indexOf = **function** (obj, start) {
497. **for** (**var** i = (start || 0), j = **this**.length; i < j; i++) {
498. **if** (**this**[i] === obj) {
499. **return** i;
500. }
501. }
502. **return** -1;
503. }
504. }
506. **if** (!Array.prototype.lastIndexOf) {
507. Array.prototype.indexOf = **function** (obj, start) {
508. **for** (**var** i = (start - 1 || **this**.length - 1), j = -1; i > j; i--) {
509. **if** (**this**[i] === obj) {
510. **return** i;
511. }
512. }
513. **return** -1;
514. }
515. }
517. String.prototype.toArray = **function** () {
518. **var** res = [];
519. **for** (**var** i = 0; i < **this**.length; i++) {
520. res.push(**this**.toString().slice(i, i + 1));
521. }
522. **return** res;
523. }
525. Array.prototype.toArray = **function** () {
526. **return** **this**;
527. }
529. String.prototype.replaceAll = **function** (search, replace) {
530. **return** **this**.split(search).join(replace);
531. }
533. String.prototype.toXMLFormat = **function** () {
534. **return** **this**.toString().replaceAll('&', '&').replaceAll('"', '"').replaceAll("'", ''').replaceAll('<', '<').replaceAll('>', '>');
535. }
537. String.prototype.toXML = **function** () {
538. **return** '<value>' + **this**.toString().toXMLFormat() + '</value>'
539. }
541. Number.prototype.getKRL = **function** () {
542. **return** (**this**).toString();
543. }
545. Number.prototype.hasObjAttrRef = **function** (n, i, d) {
546. **return** **false**;
547. }
549. String.prototype.getKRL = **function** () {
550. **return** '"' + **this**.toString() + '"';
551. }
553. Number.prototype.toXML = **function** () {
554. **return** '<value>' + **this**.toString().toXMLFormat() + '</value>'
555. }
557. String.prototype.hasObjAttrRef = **function** (n, i, d) {
558. **return** **false**;
559. }
561. Math.logb = **function** (number, base) {
562. **return** Math.log(number) / Math.log(base);
563. };
565. Math.ctan = **function** (x) {
566. **return** 1 / Math.tan(x);
567. }
569. Math.ctanh = **function** (x) {
570. **return** 1 / Math.tanh(x);
571. }
573. Math.actan = **function** (x) {
574. **return** Math.PI / 2 - Math.atan(x);
575. }
577. Math.actanh = **function** (x) {
578. **return** (Math.logb((x + 1) / (x - 1), Math.E)) \* 0.5
579. }
581. Array.prototype.includesAll = **function** (a) {
582. **var** res = **true**;
583. **for** (**var** i = 0; i < a.length; i++) {
584. res = res && (**this**.indexOf(a[i]) != -1);
585. }
586. **return** res;
587. }
589. Number.prototype.calculate = **function** () {
590. **return** parseFloat(**this**.toString());
591. }
592. XMLUtils.js
593. **var** ActiveXObject;
594. **var** XMLDom = **function** (XMLString) {
595. **this**.ActiveXMode = (ActiveXObject != undefined);
596. **if** (**this**.ActiveXMode) {
597. **var** x = **new** ActiveXObject("Microsoft.XMLDOM");
598. x.async = **false**;
599. x.loadXML(XMLString);
600. **this**.creator = x;
601. **this**.XMLControl = x.documentElement;
602. } **else** {
603. **this**.creator = **new** DOMParser().parseFromString(XMLString, 'text/xml');
604. **this**.XMLControl = **new** DOMParser().parseFromString(XMLString, 'text/xml').documentElement;
605. }
606. **this**.parentNode = **null**;
607. };
609. XMLDom.prototype.XML = **function** () {
610. **if** (**this**.ActiveXMode) {
611. **return** **this**.XMLControl.xml;
612. } **else** {
613. **return** **new** XMLSerializer().serializeToString(**this**.XMLControl);
614. }
615. }
617. XMLDom.prototype.createElement = **function** (tag) {
618. **var** element = **this**.creator.createElement(tag);
619. **if** (**this**.ActiveXMode) {
620. **return** **new** XMLDom(element.xml);
621. } **else** {
622. **return** **new** XMLDom(**new** XMLSerializer().serializeToString(element));
623. }
624. }
626. XMLDom.prototype.findParentNode = **function** (doc) {
627. **var** res = **null**;
628. **if** (doc.parentNode != undefined) {
629. **if** (doc.parentNode.setAttribute != undefined) {
630. **if** (**this**.ActiveXMode) {
631. res = **new** XMLDom(doc.parentNode.xml);
632. } **else** {
633. res = **new** XMLDom(**new** XMLSerializer().serializeToString(doc.parentNode));
634. }
635. res.parentNode = **this**.findParentNode(doc.parentNode);
636. }
637. }
638. **return** res;
639. }
641. XMLDom.prototype.appendChild = **function** (Node) {
642. **var** append = Node;
643. **if** (Node.XMLControl) {
644. append = Node.XMLControl;
645. }
646. **var** child = **this**.XMLControl.appendChild(append);
647. **var** res;
648. **if** (**this**.ActiveXMode) {
649. res = **new** XMLDom(child.xml);
650. } **else** {
651. res = **new** XMLDom(**new** XMLSerializer().serializeToString(child));
652. }
653. res.parentNode = **this**;
654. **return** res;
655. }
657. XMLDom.prototype.setText = **function** (text) {
658. **if** (**this**.ActiveXMode) {
659. **this**.XMLControl.text = text;
660. } **else** {
661. **this**.XMLControl.textContent = text;
662. }
663. }
665. XMLDom.prototype.getText = **function** () {
666. **if** (**this**.ActiveXMode) {
667. **return** **this**.XMLControl.text;
668. } **else** {
669. **return** **this**.XMLControl.textContent;
670. }
671. }
673. XMLDom.prototype.getChildNodes = **function** () {
674. **var** els = **this**.XMLControl.childNodes;
675. **var** res = [];
676. **if** (**this**.ActiveXMode) {
677. **for** (**var** i = 0; i < els.length; i++) {
678. **var** r = **new** XMLDom(els[i].xml);
679. r.parentNode = **this**;
680. res.push(r)
681. }
682. } **else** {
683. **for** (**var** i = 0; i < els.length; i++) {
684. **var** r = **new** XMLDom(**new** XMLSerializer().serializeToString(els[i]));
685. r.parentNode = **this**;
686. res.push(r)
687. }
688. }
689. **return** res;
690. }
692. XMLDom.prototype.getElementsByClassName = **function** (name) {
693. **var** els = **this**.getChildNodes()
694. **var** res = [];
695. **for** (**var** i = 0; i < els.length; i++) {
696. **if** (els[i].getAttribute('class') == name) {
697. res.push(els[i]);
698. res = res.concat(els[i].getElementsByClassName(name));
699. }
700. }
701. **return** res;
702. }
704. XMLDom.prototype.getElementsByTagName = **function** (name) {
705. **var** els = **this**.getChildNodes()
706. **var** res = [];
707. **for** (**var** i = 0; i < els.length; i++) {
708. **if** (els[i].XMLControl.tagName == name) {
709. res.push(els[i]);
710. res = res.concat(els[i].getElementsByTagName(name));
711. }
712. }
713. **return** res;
714. }
716. XMLDom.prototype.getElementById = **function** (id) {
717. **for** (**var** i = 0; i < **this**.getChildNodes().length; i++) {
718. **if** (**this**.getChildNodes()[i].getAttribute('id') == id) {
719. **return** **this**.getChildNodes()[i];
720. } **else** {
721. **return** **this**.getChildNodes()[i].getElementById(id)
722. }
723. }
724. **return** **null**;
725. }
727. XMLDom.prototype.setAttribute = **function** (name, value) {
728. **this**.XMLControl.setAttribute(name, value);
729. }
730. XMLDom.prototype.getAttribute = **function** (name) {
731. **return** **this**.XMLControl.getAttribute(name);
732. }
734. XMLDom.prototype.removeAttribute = **function** (name) {
735. **this**.XMLControl.removeAttribute(name);
736. }
737. AT\_KRL\_Editor.js
738. **var** AT\_KRL\_Editor = **function** () {
739. **this**.types = [];
740. **this**.objects = [];
741. **this**.rules = [];
742. }
744. AT\_KRL\_Editor.prototype.clear = **function** () {
745. **this**.types = [];
746. **this**.objects = [];
747. **this**.rules = [];
748. }
750. AT\_KRL\_Editor.prototype.getAllTypes = **function** () {
751. **return** [{
752. "id": 0,
753. "name": ["ЧИСЛО"],
754. "meta": "number",
755. "type": "number"
756. }, {
757. "id": 1,
758. "name": ["СИМВОЛ"],
759. "meta": "string",
760. "type": "string"
761. }, {
762. "id": 2,
763. "name": ["СИМВОЛ", "НЕЧЕТКИЙ"],
764. "meta": "fuzzy",
765. "type": "string"
766. }];
767. }
769. AT\_KRL\_Editor.prototype.verificateType = **function** (t) {
770. **if** (t && t.vType != **null**) {
771. **try** {
772. **switch** (t.vType) {
773. **case** 0:
774. **return** (t.values && t.values.length && t.values.length == 2 && !isNaN(parseFloat(t.values[0])) && !isNaN(parseFloat(t.values[1])) && parseFloat(t.values[0]) == t.values[0] && parseFloat(t.values[1]) == t.values[1] && t.values[1] >= t.values[0]);
775. **case** 1:
776. **var** res = (t.values && t.values.length)
777. **if** (res) {
778. **for** (**var** i = 0; i < t.values.length; i++) {
779. res = res && **typeof** (t.values[i]) == "string";
780. **if** (!res) {
781. **break**;
782. }
783. }
784. }
785. **return** res;
786. **case** 2:
787. **var** res = (t.values && t.values.length)
788. **if** (res) {
789. **for** (**var** i = 0; i < t.values.length; i++) {
790. res = res && **typeof** (t.values[i]) == "string";
791. **if** (!res) {
792. **break**;
793. }
794. }
795. }
796. **var** FP = [];
797. res = res && (t.FP.length == t.values.length)
798. **for** (**var** i = 0; i < t.FP.length; i++) {
799. FP.push(t.FP[i].value);
800. **var** min = t.FP[i].min;
801. **var** max = t.FP[i].max;
802. **for** (**var** j = 0; j < t.FP[i].coordinates.length; j++) {
803. res = res && (t.FP[i].coordinates[j].X >= min && t.FP[i].coordinates[j].X <= max && t.FP[i].coordinates[j].Y >= 0 && t.FP[i].coordinates[j].Y <= 1)
804. }
805. **if** (!res) {
806. **break**;
807. }
808. }
809. **for** (**var** i = 0; i < t.values.length; i++) {
810. **if** (!res) {
811. **break**;
812. }
813. res = res && FP.indexOf(t.values[i] != -1);
814. }
815. **return** res;
816. **default**:
817. **return** **false**;
818. }
820. } **catch** (e) {
821. **return** **false**;
822. }
823. } **else** {
824. **return** **false**;
825. }
826. }
828. AT\_KRL\_Editor.prototype.pushType = **function** (t) {
829. **var** names = [];
830. **for** (**var** i = 0; i < **this**.types.length; i++) {
831. names.push(**this**.types[i].name);
832. }
833. **if** (names.indexOf(t.name) != -1) {
834. t.name = "ТИП" + **this**.types.length;
835. }
836. **this**.types.push(t);
837. }
839. AT\_KRL\_Editor.prototype.pushObject = **function** (o) {
840. **var** names = [];
841. **for** (**var** i = 0; i < **this**.objects.length; i++) {
842. names.push(**this**.objects[i].name);
843. }
844. **if** (names.indexOf(o.name) != -1) {
845. o.name = "ОБЪЕКТ" + **this**.objects.length;
846. }
847. **this**.objects.push(o);
848. }
850. AT\_KRL\_Editor.prototype.pushRule = **function** (r) {
851. **var** names = [];
852. **for** (**var** i = 0; i < **this**.rules.length; i++) {
853. names.push(**this**.rules[i].name);
854. }
855. **if** (names.indexOf(r.name) != -1) {
856. r.name = "ПРАВИЛО" + **this**.rules.length;
857. }
858. **this**.rules.push(r);
859. }
861. AT\_KRL\_Editor.prototype.createType = **function** (a, t, v, n) {
862. **var** type = **new** AT\_KRL\_Type(a, t, v, n, **this**);
863. **return** type;
864. }
866. AT\_KRL\_Editor.prototype.createObject = **function** (a, as, n) {
867. **var** o = **new** AT\_KRL\_Type(a, as, n, **this**);
868. **return** o;
869. }
871. AT\_KRL\_Editor.prototype.getKRL = **function** () {
872. **var** res = "";
873. **for** (**var** i = 0; i < **this**.types.length; i++) {
874. res += **this**.types[i].getKRL();
875. }
877. **for** (**var** i = 0; i < **this**.objects.length; i++) {
878. res += **this**.objects[i].getKRL();
879. }
881. **for** (**var** i = 0; i < **this**.rules.length; i++) {
882. res += **this**.rules[i].getKRL();
883. }
884. **return** res;
885. }
887. AT\_KRL\_Editor.prototype.toXML = **function** () {
888. **var** res = '<?xml version="1.0" encoding="UTF-8"?><knowledge-base creation-date="' + (**new** Date()).toPString() + '"><problem-info></problem-info><types>';
889. **for** (**var** i = 0; i < **this**.types.length; i++) {
890. res += **this**.types[i].toXML();
891. }
893. res += '</types>' + **this**.classXML();
894. **return** res + "</knowledge-base>";
895. }
897. AT\_KRL\_Editor.prototype.classXML = **function** () {
898. **var** classes = **new** XMLDom('<classes/>');
899. **var** world = **new** XMLDom('<class id="world" desc="Класс верхнего уровня, включающий в себя экземпляры других классов и общие правила" />');
900. **var** properties = **new** XMLDom('<properties/>');
901. **for** (**var** i = 0; i < **this**.objects.length; i++) {
902. classes.appendChild(**new** XMLDom(**this**.objects[i].toXML(i)));
903. properties.appendChild(**new** XMLDom('<property id="' + **this**.objects[i].name + '" type="КЛАСС' + (i+1).toString() + '" desc="экземпляр класса КЛАСС' + (i+1).toString() + '" />'));
904. }
905. world.appendChild(properties);
906. world.appendChild(**new** XMLDom(**this**.rulesXML()));
907. world.appendChild(**new** XMLDom('<methods />'))
908. classes.appendChild(world);
909. **return** classes.XML();
910. }
912. AT\_KRL\_Editor.prototype.rulesXML = **function**(){
913. **var** rs = **new** XMLDom('<rules />');
914. **for** (**var** i = 0; i < **this**.rules.length; i++){
915. rs.appendChild(**new** XMLDom(**this**.rules[i].toXML(i)));
916. }
917. **return** rs.XML();
918. }
919. AT\_KRL\_Type.js
920. **var** AT\_KRL\_Type = **function** (a, t, v, n, e) {
921. **function** validateName(s) {
922. **var** inv = '~!@#$%^&\*()-=+/\*|\\?.,:;`\'" ';
923. **for** (**var** i = 0; i < inv.length; i++) {
924. **if** (s.indexOf(inv[i]) != -1) {
925. **throw** **new** Error('Invalid character "' + inv[i] + '" in name ' + s);
926. }
927. }
928. **return** s;
929. }
930. **this**.name = validateName(a ? (a.replaceAll(' ', '')) : ("ТИП" + (e ? (e.types.length + 1) : 1)));
931. **this**.vType = t;
932. **this**.values = v.values;
933. **this**.comment = n || **this**.name;
934. **if** (t == 2) {
935. **this**.FP = v.FP;
936. }
937. **if** (e && e.verificateType(**this**)) {
938. e.pushType(**this**)
939. } **else** **if** (e) {
940. **this**.vType = **null**;
941. **this**.values = **null**;
942. **throw** **new** Error('Mismatch between type of values and values');
943. } **else** {
944. **var** e = **new** AT\_KRL\_Editor();
945. **if** (!e.verificateType(**this**)) {
946. **this**.vType = **null**;
947. **this**.values = **null**;
948. **throw** **new** Error('Mismatch between type of values and values');
949. }
950. }
951. }
953. AT\_KRL\_Type.prototype.getAllTypes = **function** () {
954. **return** (**new** AT\_KRL\_Editor()).getAllTypes();
955. }
957. AT\_KRL\_Type.prototype.getFPKRL = **function** (f) {
958. **var** res = '"' + f.value + '" ' + f.min + ' ' + f.max + ' ' + f.coordinates.length + ' ={';
959. **for** (**var** i = 0; i < f.coordinates.length; i++) {
960. res += f.coordinates[i].X.toString() + '|' + f.coordinates[i].Y.toString() + ((i < (f.coordinates.length - 1)) ? '; ' : '}');
961. }
962. **return** res;
963. }
965. AT\_KRL\_Type.prototype.getKRL = **function** () {
966. **var** res = "ТИП " + **this**.name + "\n" + **this**.getAllTypes()[**this**.vType].name[0] + "\n";
967. **switch** (**this**.vType) {
968. **case** 0:
969. res += "ОТ " + **this**.values[0] + "\nДО " + **this**.values[1] + "\n";
970. **break**;
971. **case** 1:
972. **for** (**var** i = 0; i < **this**.values.length; i++) {
973. res += '"' + **this**.values[i] + '"\n';
974. }
975. **break**;
976. **case** 2:
977. **for** (**var** i = 0; i < **this**.values.length; i++) {
978. res += '"' + **this**.values[i] + '"\n';
979. }
981. res += 'НЕЧЕТКИЙ\n' + **this**.FP.length + "\n";
982. **for** (**var** i = 0; i < **this**.FP.length; i++) {
983. res += **this**.getFPKRL(**this**.FP[i]) + '\n';
984. }
985. **break**;
986. }
987. res += "КОММЕНТАРИЙ " + **this**.comment + "\n\n";
988. **return** res;
989. }
991. AT\_KRL\_Type.prototype.isValidValue = **function** (v) {
992. **if** (**typeof** (v) == **this**.getAllTypes()[**this**.vType].type) {
993. **var** t = **this**.vType;
994. **switch** (t) {
995. **case** 0:
996. **return** (v >= **this**.values[0] && v <= **this**.values[1]);
997. **break**;
998. **case** 1:
999. **return** (**this**.values.indexOf(v) != -1);
1000. **break**;
1001. **case** 2:
1002. **return** (**typeof** (v) == "number" || **this**.values.indexOf(v) != -1);
1003. **break**;
1004. **default**:
1005. **return** **false**;
1006. }
1007. }
1008. **return** **false**;
1009. }
1011. AT\_KRL\_Type.prototype.toXML = **function** () {
1012. **var** Doc = **new** XMLDom('<type/>');
1013. Doc.setAttribute('id', **this**.name)
1014. Doc.setAttribute('meta', **this**.getAllTypes()[**this**.vType].meta);
1015. Doc.setAttribute('desc', **this**.comment);
1016. **var** t = **this**.vType;
1017. **switch** (t) {
1018. **case** 0:
1019. **var** from = Doc.createElement('from');
1020. from.setText(**this**.values[0].toString())
1021. Doc.appendChild(from);
1022. **var** to = Doc.createElement('to');
1023. to.setText(**this**.values[1].toString());
1024. Doc.appendChild(to);
1025. **break**;
1026. **case** 1:
1027. **for** (**var** i = 0; i < **this**.values.length; i++) {
1028. **var** value = Doc.createElement('value');
1029. value.setText(**this**.values[i]);
1030. Doc.appendChild(value);
1031. }
1032. **break**;
1033. **case** 2:
1034. **for** (**var** i = 0; i < **this**.FP.length; i++) {
1035. **var** parametr = Doc.createElement('parametr')
1036. **var** value = Doc.createElement('value');
1037. parametr.setAttribute('min-value',**this**.FP[i].min.toString());
1038. parametr.setAttribute('max-value',**this**.FP[i].max.toString());
1039. value.setText(**this**.FP[i].value);
1040. **var** coordsXML = Doc.createElement('mf');
1041. **for** (**var** j = 0; j < **this**.FP[i].coordinates.length; j++){
1042. **var** pt = Doc.createElement('point');
1043. pt.setAttribute('X',**this**.FP[i].coordinates[j].X.toString());
1044. pt.setAttribute('Y',**this**.FP[i].coordinates[j].Y.toString());
1045. coordsXML.appendChild(pt);
1046. }
1047. parametr.appendChild(value);
1048. parametr.appendChild(coordsXML)
1049. Doc.appendChild(parametr);
1050. }
1051. **break**;
1052. }
1053. **return** Doc.XML();
1054. }
1055. AT\_KRL\_Object.js
1056. **var** AT\_KRL\_Object = **function** (a, as, n, e) {
1057. **function** validateName(s) {
1058. **var** inv = '~!@#$%^&\*()-=+/\*|\\?.,:;`\'" ';
1059. **for** (**var** i = 0; i < inv.length; i++) {
1060. **if** (s.indexOf(inv[i]) != -1) {
1061. **throw** **new** Error('Invalid character "' + inv[i] + '" in name ' + s);
1062. }
1063. }
1064. **return** s;
1065. }
1067. **this**.attributes = as || [];
1068. **this**.name = validateName((**typeof** (a) == "string" ? a.replaceAll(' ', '') : ("ОБЪЕКТ" + ((e && e.objects) ? e.objects.length : 1))));
1069. **this**.comment = n || **this**.name;
1070. **if** (e) {
1071. e.pushObject(**this**);
1072. **this**.editor = e;
1073. }
1074. }
1076. AT\_KRL\_Object.prototype.verificateAttribute = **function** (a) {
1077. **try** {
1078. **var** v = a.value;
1079. **var** t = a.type.vType;
1080. **var** vs = a.type.values;
1081. **switch** (t) {
1082. **case** 0:
1083. **return** (**typeof** (v) == "number" && v >= vs[0] && v <= vs[1]);
1084. **break**;
1085. **case** 1:
1086. **return** (**typeof** (v) == "string" && vs.indexOf(v) != -1);
1087. **break**;
1088. **case** 2:
1089. **return** **true**;
1090. **break**;
1091. }
1092. **return** **false**;
1093. } **catch** (e) {
1094. **return** **false**;
1095. }
1096. }
1098. AT\_KRL\_Object.prototype.pushAttribute = **function** (a) {
1099. **var** names = [];
1100. **for** (**var** i = 0; i < **this**.attributes.length; i++) {
1101. names.push(**this**.attributes[i].name);
1102. }
1103. **if** (names.indexOf(a.name) != -1) {
1104. a.name = "АТРИБУТ" + names.length;
1105. }
1106. **this**.attributes.push(a);
1107. }
1109. AT\_KRL\_Object.prototype.createAttribute = **function** (a, t, v, n) {
1110. **var** a = **new** AT\_KRL\_Attribute(a, t, v, n, **this**);
1111. **return** a;
1112. }
1114. AT\_KRL\_Object.prototype.getKRL = **function** () {
1115. **var** res = "ОБЪЕКТ " + **this**.name + "\nГРУППА ГРУППА1\nАТРИБУТЫ\n";
1116. **for** (i = 0; i < **this**.attributes.length; i++) {
1117. res += **this**.attributes[i].getKRL();
1118. }
1119. res += "КОММЕНТАРИЙ " + **this**.comment + "\n\n";
1120. **return** res;
1121. }
1123. **var** AT\_KRL\_Attribute = **function** (a, t, v, n, o) {
1124. **function** validateName(s) {
1125. **var** inv = '~!@#$%^&\*()-=+/\*|\\?.,:;`\'" ';
1126. **for** (**var** i = 0; i < inv.length; i++) {
1127. **if** (s.indexOf(inv[i]) != -1) {
1128. **throw** **new** Error('Invalid character "' + inv[i] + '" in name ' + s);
1129. }
1130. }
1131. **return** s;
1132. }
1133. **this**.name = validateName(**typeof** (a) == "string" ? a.replaceAll(' ', '') : ("АТРИБУТ" + (o && o.attributes ? o.attributes.length : 1)));
1134. **if** (!t) {
1135. **throw** **new** Error('Wrong type parameter');
1136. }
1137. **this**.type = t;
1138. **this**.value = v;
1139. **this**.comment = n || **this**.name;
1140. **if** (o && o.verificateAttribute(**this**)) {
1141. o.pushAttribute(**this**)
1142. } **else** **if** (o) {
1143. **this**.type = **null**;
1144. **this**.value = **null**;
1145. **throw** **new** Error('Mismatch between type of values and values');
1146. } **else** {
1147. **var** o = **new** AT\_KRL\_Object()
1148. **if** (!o.verificateAttribute(**this**)) {
1149. **this**.type = **null**;
1150. **this**.value = **null**;
1151. **throw** **new** Error('Mismatch between type of values and values');
1152. }
1153. }
1154. }
1156. AT\_KRL\_Attribute.prototype.getKRL = **function** () {
1157. **var** res = "АТРИБУТ " + **this**.name + "\nТИП " + **this**.type.name + "\nКОММЕНТАРИЙ " + **this**.comment + "\n";
1158. **return** res;
1159. }
1161. AT\_KRL\_Object.prototype.toXML = **function** (num) {
1162. **var** index = (num + 1) || 1;
1163. **var** cls = **new** XMLDom('<class/>');
1164. cls.setAttribute('id', 'КЛАСС' + index.toString());
1165. cls.setAttribute('desc', **this**.comment);
1166. **var** properties = **new** XMLDom('<properties />');
1167. **for** (**var** i = 0; i < **this**.attributes.length; i++) {
1168. **var** property = **new** XMLDom('<property/>');
1169. property.setAttribute('id', **this**.attributes[i].name);
1170. property.setAttribute('type', **this**.attributes[i].type.name);
1171. property.setAttribute('desc', **this**.attributes[i].comment);
1172. property.setAttribute('source', **this**.attributeSourceType(i));
1173. **if** (**this**.attributeSourceType(i) == 'question') {
1174. **var** question = property.createElement('question');
1175. question.setText(**this**.attributes[i].comment);
1176. property.appendChild(question);
1177. }
1178. properties.appendChild(property);
1179. }
1180. cls.appendChild(properties);
1181. **return** cls.XML();
1182. }
1184. AT\_KRL\_Object.prototype.attributeSourceType = **function** (index) {
1185. **if** (**this**.editor) {
1186. **var** e = **this**.editor;
1187. **for** (**var** i = 0; i < e.rules.length; i++) {
1188. **if** (e.rules[i].hasObjAttrRef(**this**.name, index, **false**)) {
1189. **return** "inferred";
1190. }
1191. }
1192. }
1193. **return** "question";
1194. }
1195. AT\_KRL\_Expressions.js
1196. **var** AT\_KRL\_StFzExpression = **function** (object, aIndex) {
1197. **if** (object.attributes[aIndex].type.vType != 1 && object.attributes[aIndex].type.vType != 2) {
1198. **throw** **new** Error('Type of attributes must be string or fuzzy');
1199. }
1200. **this**.expressions = [{}];
1201. **this**.expressions[0].object = object;
1202. **this**.expressions[0].aIndex = aIndex;
1203. }
1205. AT\_KRL\_StFzExpression.prototype.getKRL = **function** () {
1206. **return** (**this**.expressions[0].object.name) + '.' + (**this**.expressions[0].object.attributes[**this**.expressions[0].aIndex].name);
1207. }
1209. AT\_KRL\_StFzExpression.prototype.hasObjAttrRef = **function** (name, index, deep) {
1210. **return** (**this**.expressions[0].object.name == name && **this**.expressions[0].aIndex == index);
1211. }
1213. AT\_KRL\_StFzExpression.prototype.toXML = **function** () {
1214. **var** r1 = **new** XMLDom('<ref />')
1215. r1.setAttribute('id', **this**.expressions[0].object.name);
1216. **var** r2 = **new** XMLDom('<ref />');
1217. r2.setAttribute('id', **this**.expressions[0].object.attributes[**this**.expressions[0].aIndex].name);
1218. r1.appendChild(r2);
1219. **return** r1.XML();
1220. }
1222. **var** AT\_KRL\_MathExpression = **function** (es, sing) {
1223. **function** validateEs(es, sing, t) {
1224. **var** res = **true**;
1225. **for** (**var** i = 0; i < es.length; i++) {
1226. res = res && t.isNumber(es[i]);
1227. }
1228. **if** (!res) {
1229. **throw** **new** Error(JSON.stringify(es[i]) + ' must be able to be converted to number type');
1230. }
1232. **var** sgs = t.getAllSings();
1233. **var** sg;
1234. **for** (**var** i = 0; i < sgs.length; i++) {
1235. **if** (sgs[i].sing == sing) {
1236. sg = sgs[i];
1237. }
1238. }
1240. **if** (es.length > sg.pos) {
1241. **throw** **new** Error('Too much elements for making "' + sing + '" expression');
1242. }
1243. **if** (sg.type.indexOf('bin') != -1 && es.length == 1 && sg.type.indexOf('coef') == -1) {
1244. **throw** **new** Error('Too little elements for making "' + sing + '" expression');
1245. }
1246. **return** es;
1247. }
1249. **function** validateSing(s, t) {
1250. **var** sgs = t.getAllSings();
1251. **var** res = **false**;
1252. **for** (**var** i = 0; i < sgs.length; i++) {
1253. res = res || sgs[i].sing == s;
1254. **if** (res) {
1255. **return** s;
1256. }
1257. }
1258. **if** (!res) {
1259. **throw** **new** Error('Invalid sing ' + JSON.stringify(s))
1260. }
1261. }
1263. **this**.sing = validateSing(sing, **this**);
1264. **this**.expressions = es.length ? validateEs(es, sing, **this**) : validateEs([es], sing, **this**);
1265. }
1267. AT\_KRL\_MathExpression.prototype.isNumber = **function** (ex) {
1268. **if** (**typeof** (ex) == "number") {
1269. **return** **true**;
1270. }
1271. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && ex.object.attributes[ex.aIndex].type.vType == 0) {
1272. **return** **true**;
1273. }
1274. **var** res = (ex.expressions) ? **true** : **false**;
1275. **if** (res) {
1276. **for** (**var** i = 0; i < ex.expressions.length; i++) {
1277. res = res && **this**.isNumber(ex.expressions[i]);
1278. }
1279. }
1280. **return** res;
1281. }
1283. AT\_KRL\_MathExpression.prototype.getAllSings = **function** () {
1284. **return** [{
1285. "sing": **null**,
1286. "name": **null**,
1287. "pos": 2,
1288. "type": [],
1289. "priority": **null**,
1290. "calc": **function** (es) {
1291. **function** getExpressionValue(ex) {
1292. **if** (**typeof** (ex) == "number") {
1293. **return** ex;
1294. }
1295. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1296. **return** ex.object.attributes[ex.aIndex].value;
1297. }
1298. **if** (ex.expressions) {
1299. **return** ex.calculate();
1300. }
1301. }
1302. **if** (es.length == 1) {
1303. **return** getExpressionValue(es[0]);
1304. }
1305. }
1306. }, {
1307. "sing": "-",
1308. "name": "minus",
1309. "pos": 2,
1310. "type": ["bin", "coef"],
1311. "priority": [0],
1312. "calc": **function** (es) {
1313. **function** getExpressionValue(ex) {
1314. **if** (**typeof** (ex) == "number") {
1315. **return** ex;
1316. }
1317. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1318. **return** ex.object.attributes[ex.aIndex].value;
1319. }
1320. **if** (ex.expressions) {
1321. **return** ex.calculate();
1322. }
1323. }
1324. **if** (es.length == 1) {
1325. **return** -getExpressionValue(es[0]);
1326. }
1327. **if** (es.length == 2) {
1328. **return** getExpressionValue(es[0]) - getExpressionValue(es[1]);
1329. }
1330. }
1331. }, {
1332. "sing": "+",
1333. "name": "plus",
1334. "pos": Infinity,
1335. "type": ["bin"],
1336. "priority": [0],
1337. "calc": **function** (es) {
1338. **function** getExpressionValue(ex) {
1339. **if** (**typeof** (ex) == "number") {
1340. **return** ex;
1341. }
1342. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1343. **return** ex.object.attributes[ex.aIndex].value;
1344. }
1345. **if** (ex.expressions) {
1346. **return** ex.calculate();
1347. }
1348. }
1349. **var** res = 0;
1350. **for** (**var** i = 0; i < es.length; i++) {
1351. res += getExpressionValue(es[i]);
1352. }
1353. **return** res;
1354. }
1355. }, {
1356. "sing": "/",
1357. "name": "dev",
1358. "pos": 2,
1359. "type": ["bin"],
1360. "priority": [1],
1361. "calc": **function** (es) {
1362. **function** getExpressionValue(ex) {
1363. **if** (**typeof** (ex) == "number") {
1364. **return** ex;
1365. }
1366. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1367. **return** ex.object.attributes[ex.aIndex].value;
1368. }
1369. **if** (ex.expressions) {
1370. **return** ex.calculate();
1371. }
1372. }
1373. **if** (es.length == 2) {
1374. **return** getExpressionValue(es[0]) / getExpressionValue(es[1]);
1375. }
1376. }
1377. }, {
1378. "sing": "\*",
1379. "name": "mult",
1380. "pos": Infinity,
1381. "type": ["bin"],
1382. "priority": [1],
1383. "calc": **function** (es) {
1384. **function** getExpressionValue(ex) {
1385. **if** (**typeof** (ex) == "number") {
1386. **return** ex;
1387. }
1388. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1389. **return** ex.object.attributes[ex.aIndex].value;
1390. }
1391. **if** (ex.expressions) {
1392. **return** ex.calculate();
1393. }
1394. }
1395. **var** res = 1;
1396. **for** (**var** i = 0; i < es.length; i++) {
1397. res = res \* getExpressionValue(es[i]);
1398. }
1399. **return** res;
1400. }
1401. }, {
1402. "sing": "^",
1403. "name": "pow",
1404. "pos": 2,
1405. "type": ["bin"],
1406. "priority": [1],
1407. "calc": **function** (es) {
1408. **function** getExpressionValue(ex) {
1409. **if** (**typeof** (ex) == "number") {
1410. **return** ex;
1411. }
1412. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1413. **return** ex.object.attributes[ex.aIndex].value;
1414. }
1415. **if** (ex.expressions) {
1416. **return** ex.calculate();
1417. }
1418. }
1419. **if** (es.length == 2) {
1420. **return** Math.pow(getExpressionValue(es[0]), getExpressionValue(es[1]));
1421. }
1422. }
1423. }, {
1424. "sing": "sin",
1425. "name": "sin",
1426. "pos": 1,
1427. "type": ["op"],
1428. "priority": [2],
1429. "calc": **function** (es) {
1430. **function** getExpressionValue(ex) {
1431. **if** (**typeof** (ex) == "number") {
1432. **return** ex;
1433. }
1434. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1435. **return** ex.object.attributes[ex.aIndex].value;
1436. }
1437. **if** (ex.expressions) {
1438. **return** ex.calculate();
1439. }
1440. }
1441. **if** (es.length == 1) {
1442. **return** Math.sin(getExpressionValue(es[0]));
1443. }
1444. }
1445. }, {
1446. "sing": "cos",
1447. "name": "cos",
1448. "pos": 1,
1449. "type": ["op"],
1450. "priority": [2],
1451. "calc": **function** (es) {
1452. **function** getExpressionValue(ex) {
1453. **if** (**typeof** (ex) == "number") {
1454. **return** ex;
1455. }
1456. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1457. **return** ex.object.attributes[ex.aIndex].value;
1458. }
1459. **if** (ex.expressions) {
1460. **return** ex.calculate();
1461. }
1462. }
1463. **if** (es.length == 1) {
1464. **return** Math.cos(getExpressionValue(es[0]));
1465. }
1466. }
1467. }, {
1468. "sing": "tan",
1469. "name": "tan",
1470. "pos": 1,
1471. "type": ["op"],
1472. "priority": [2],
1473. "calc": **function** (es) {
1474. **function** getExpressionValue(ex) {
1475. **if** (**typeof** (ex) == "number") {
1476. **return** ex;
1477. }
1478. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1479. **return** ex.object.attributes[ex.aIndex].value;
1480. }
1481. **if** (ex.expressions) {
1482. **return** ex.calculate();
1483. }
1484. }
1485. **if** (es.length == 1) {
1486. **return** Math.tan(getExpressionValue(es[0]));
1487. }
1488. }
1489. }, {
1490. "sing": "ctan",
1491. "name": "ctan",
1492. "pos": 1,
1493. "type": ["op"],
1494. "priority": [2],
1495. "calc": **function** (es) {
1496. **function** getExpressionValue(ex) {
1497. **if** (**typeof** (ex) == "number") {
1498. **return** ex;
1499. }
1500. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1501. **return** ex.object.attributes[ex.aIndex].value;
1502. }
1503. **if** (ex.expressions) {
1504. **return** ex.calculate();
1505. }
1506. }
1507. **if** (es.length == 1) {
1508. **return** Math.ctan(getExpressionValue(es[0]));
1509. }
1510. }
1511. }, {
1512. "sing": "asin",
1513. "name": "asin",
1514. "pos": 1,
1515. "type": ["op"],
1516. "priority": [2],
1517. "calc": **function** (es) {
1518. **function** getExpressionValue(ex) {
1519. **if** (**typeof** (ex) == "number") {
1520. **return** ex;
1521. }
1522. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1523. **return** ex.object.attributes[ex.aIndex].value;
1524. }
1525. **if** (ex.expressions) {
1526. **return** ex.calculate();
1527. }
1528. }
1529. **if** (es.length == 1) {
1530. **return** Math.asin(getExpressionValue(es[0]));
1531. }
1532. }
1533. }, {
1534. "sing": "acos",
1535. "name": "acos",
1536. "pos": 1,
1537. "type": ["op"],
1538. "priority": [2],
1539. "calc": **function** (es) {
1540. **function** getExpressionValue(ex) {
1541. **if** (**typeof** (ex) == "number") {
1542. **return** ex;
1543. }
1544. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1545. **return** ex.object.attributes[ex.aIndex].value;
1546. }
1547. **if** (ex.expressions) {
1548. **return** ex.calculate();
1549. }
1550. }
1551. **if** (es.length == 1) {
1552. **return** Math.acos(getExpressionValue(es[0]));
1553. }
1554. }
1555. }, {
1556. "sing": "atan",
1557. "name": "atan",
1558. "pos": 1,
1559. "type": ["op"],
1560. "priority": [2],
1561. "calc": **function** (es) {
1562. **function** getExpressionValue(ex) {
1563. **if** (**typeof** (ex) == "number") {
1564. **return** ex;
1565. }
1566. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1567. **return** ex.object.attributes[ex.aIndex].value;
1568. }
1569. **if** (ex.expressions) {
1570. **return** ex.calculate();
1571. }
1572. }
1573. **if** (es.length == 1) {
1574. **return** Math.atan(getExpressionValue(es[0]));
1575. }
1576. }
1577. }, {
1578. "sing": "actan",
1579. "name": "actan",
1580. "pos": 1,
1581. "type": ["op"],
1582. "priority": [2],
1583. "calc": **function** (es) {
1584. **function** getExpressionValue(ex) {
1585. **if** (**typeof** (ex) == "number") {
1586. **return** ex;
1587. }
1588. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1589. **return** ex.object.attributes[ex.aIndex].value;
1590. }
1591. **if** (ex.expressions) {
1592. **return** ex.calculate();
1593. }
1594. }
1595. **if** (es.length == 1) {
1596. **return** Math.actan(getExpressionValue(es[0]));
1597. }
1598. }
1599. }, {
1600. "sing": "sinh",
1601. "name": "sinh",
1602. "pos": 1,
1603. "type": ["op"],
1604. "priority": [2],
1605. "calc": **function** (es) {
1606. **function** getExpressionValue(ex) {
1607. **if** (**typeof** (ex) == "number") {
1608. **return** ex;
1609. }
1610. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1611. **return** ex.object.attributes[ex.aIndex].value;
1612. }
1613. **if** (ex.expressions) {
1614. **return** ex.calculate();
1615. }
1616. }
1617. **if** (es.length == 1) {
1618. **return** Math.sinh(getExpressionValue(es[0]));
1619. }
1620. }
1621. }, {
1622. "sing": "cosh",
1623. "name": "cosh",
1624. "pos": 1,
1625. "type": ["op"],
1626. "priority": [2],
1627. "calc": **function** (es) {
1628. **function** getExpressionValue(ex) {
1629. **if** (**typeof** (ex) == "number") {
1630. **return** ex;
1631. }
1632. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1633. **return** ex.object.attributes[ex.aIndex].value;
1634. }
1635. **if** (ex.expressions) {
1636. **return** ex.calculate();
1637. }
1638. }
1639. **if** (es.length == 1) {
1640. **return** Math.cosh(getExpressionValue(es[0]));
1641. }
1642. }
1643. }, {
1644. "sing": "tanh",
1645. "name": "tanh",
1646. "pos": 1,
1647. "type": ["op"],
1648. "priority": [2],
1649. "calc": **function** (es) {
1650. **function** getExpressionValue(ex) {
1651. **if** (**typeof** (ex) == "number") {
1652. **return** ex;
1653. }
1654. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1655. **return** ex.object.attributes[ex.aIndex].value;
1656. }
1657. **if** (ex.expressions) {
1658. **return** ex.calculate();
1659. }
1660. }
1661. **if** (es.length == 1) {
1662. **return** Math.tanh(getExpressionValue(es[0]));
1663. }
1664. }
1665. }, {
1666. "sing": "ctanh",
1667. "name": "ctanh",
1668. "pos": 1,
1669. "type": ["op"],
1670. "priority": [2],
1671. "calc": **function** (es) {
1672. **function** getExpressionValue(ex) {
1673. **if** (**typeof** (ex) == "number") {
1674. **return** ex;
1675. }
1676. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1677. **return** ex.object.attributes[ex.aIndex].value;
1678. }
1679. **if** (ex.expressions) {
1680. **return** ex.calculate();
1681. }
1682. }
1683. **if** (es.length == 1) {
1684. **return** Math.ctanh(getExpressionValue(es[0]));
1685. }
1686. }
1687. }, {
1688. "sing": "asinh",
1689. "name": "asinh",
1690. "pos": 1,
1691. "type": ["op"],
1692. "priority": [2],
1693. "calc": **function** (es) {
1694. **function** getExpressionValue(ex) {
1695. **if** (**typeof** (ex) == "number") {
1696. **return** ex;
1697. }
1698. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1699. **return** ex.object.attributes[ex.aIndex].value;
1700. }
1701. **if** (ex.expressions) {
1702. **return** ex.calculate();
1703. }
1704. }
1705. **if** (es.length == 1) {
1706. **return** Math.asinh(getExpressionValue(es[0]));
1707. }
1708. }
1709. }, {
1710. "sing": "acosh",
1711. "name": "acosh",
1712. "pos": 1,
1713. "type": ["op"],
1714. "priority": [2],
1715. "calc": **function** (es) {
1716. **function** getExpressionValue(ex) {
1717. **if** (**typeof** (ex) == "number") {
1718. **return** ex;
1719. }
1720. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1721. **return** ex.object.attributes[ex.aIndex].value;
1722. }
1723. **if** (ex.expressions) {
1724. **return** ex.calculate();
1725. }
1726. }
1727. **if** (es.length == 1) {
1728. **return** Math.acosh(getExpressionValue(es[0]));
1729. }
1730. }
1731. }, {
1732. "sing": "atanh",
1733. "name": "atanh",
1734. "pos": 1,
1735. "type": ["op"],
1736. "priority": [2],
1737. "calc": **function** (es) {
1738. **function** getExpressionValue(ex) {
1739. **if** (**typeof** (ex) == "number") {
1740. **return** ex;
1741. }
1742. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1743. **return** ex.object.attributes[ex.aIndex].value;
1744. }
1745. **if** (ex.expressions) {
1746. **return** ex.calculate();
1747. }
1748. }
1749. **if** (es.length == 1) {
1750. **return** Math.atanh(getExpressionValue(es[0]));
1751. }
1752. }
1753. }, {
1754. "sing": "actanh",
1755. "name": "actanh",
1756. "pos": 1,
1757. "type": ["op"],
1758. "priority": [2],
1759. "calc": **function** (es) {
1760. **function** getExpressionValue(ex) {
1761. **if** (**typeof** (ex) == "number") {
1762. **return** ex;
1763. }
1764. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1765. **return** ex.object.attributes[ex.aIndex].value;
1766. }
1767. **if** (ex.expressions) {
1768. **return** ex.calculate();
1769. }
1770. }
1771. **if** (es.length == 1) {
1772. **return** Math.actanh(getExpressionValue(es[0]));
1773. }
1774. }
1775. }, {
1776. "sing": "exp",
1777. "name": "exp",
1778. "pos": 1,
1779. "type": ["op"],
1780. "priority": [2],
1781. "calc": **function** (es) {
1782. **function** getExpressionValue(ex) {
1783. **if** (**typeof** (ex) == "number") {
1784. **return** ex;
1785. }
1786. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1787. **return** ex.object.attributes[ex.aIndex].value;
1788. }
1789. **if** (ex.expressions) {
1790. **return** ex.calculate();
1791. }
1792. }
1793. **if** (es.length == 1) {
1794. **return** Math.exp(getExpressionValue(es[0]));
1795. }
1796. }
1797. }, {
1798. "sing": "log",
1799. "name": "log",
1800. "pos": 2,
1801. "type": ["bin", "op"],
1802. "priority": [2],
1803. "calc": **function** (es) {
1804. **function** getExpressionValue(ex) {
1805. **if** (**typeof** (ex) == "number") {
1806. **return** ex;
1807. }
1808. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1809. **return** ex.object.attributes[ex.aIndex].value;
1810. }
1811. **if** (ex.expressions) {
1812. **return** ex.calculate();
1813. }
1814. }
1815. **if** (es.length == 2) {
1816. **return** Math.logb(getExpressionValue(es[1]), getExpressionValue(es[0]));
1817. }
1818. }
1819. }, {
1820. "sing": "lg",
1821. "name": "lg",
1822. "pos": 1,
1823. "type": ["op"],
1824. "priority": [2],
1825. "calc": **function** (es) {
1826. **function** getExpressionValue(ex) {
1827. **if** (**typeof** (ex) == "number") {
1828. **return** ex;
1829. }
1830. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1831. **return** ex.object.attributes[ex.aIndex].value;
1832. }
1833. **if** (ex.expressions) {
1834. **return** ex.calculate();
1835. }
1836. }
1837. **if** (es.length == 1) {
1838. **return** Math.logb(10, getExpressionValue(es[0]));
1839. }
1840. }
1841. }, {
1842. "sing": "ln",
1843. "name": "ln",
1844. "pos": 1,
1845. "type": ["op"],
1846. "priority": [2],
1847. "calc": **function** (es) {
1848. **function** getExpressionValue(ex) {
1849. **if** (**typeof** (ex) == "number") {
1850. **return** ex;
1851. }
1852. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1853. **return** ex.object.attributes[ex.aIndex].value;
1854. }
1855. **if** (ex.expressions) {
1856. **return** ex.calculate();
1857. }
1858. }
1859. **if** (es.length == 1) {
1860. **return** Math.logb(Math.E, getExpressionValue(es[0]));
1861. }
1862. }
1863. }, {
1864. "sing": "sqrt",
1865. "name": "sqrt",
1866. "pos": 1,
1867. "type": ["op"],
1868. "priority": [2],
1869. "calc": **function** (es) {
1870. **function** getExpressionValue(ex) {
1871. **if** (**typeof** (ex) == "number") {
1872. **return** ex;
1873. }
1874. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1875. **return** ex.object.attributes[ex.aIndex].value;
1876. }
1877. **if** (ex.expressions) {
1878. **return** ex.calculate();
1879. }
1880. }
1881. **if** (es.length == 1) {
1882. **return** Math.sqrt(getExpressionValue(es[0]));
1883. }
1884. }
1885. }, {
1886. "sing": "abs",
1887. "name": "abs",
1888. "pos": 1,
1889. "type": ["op"],
1890. "priority": [2],
1891. "calc": **function** (es) {
1892. **function** getExpressionValue(ex) {
1893. **if** (**typeof** (ex) == "number") {
1894. **return** ex;
1895. }
1896. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1897. **return** ex.object.attributes[ex.aIndex].value;
1898. }
1899. **if** (ex.expressions) {
1900. **return** ex.calculate();
1901. }
1902. }
1903. **if** (es.length == 1) {
1904. **return** Math.abs(getExpressionValue(es[0]));
1905. }
1906. }
1907. }, {
1908. "sing": "sign",
1909. "name": "sign",
1910. "pos": 1,
1911. "type": ["op"],
1912. "priority": [2],
1913. "calc": **function** (es) {
1914. **function** getExpressionValue(ex) {
1915. **if** (**typeof** (ex) == "number") {
1916. **return** ex;
1917. }
1918. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1919. **return** ex.object.attributes[ex.aIndex].value;
1920. }
1921. **if** (ex.expressions) {
1922. **return** ex.calculate();
1923. }
1924. }
1925. **if** (es.length == 1) {
1926. **return** Math.sign(getExpressionValue(es[0]));
1927. }
1928. }
1929. }, {
1930. "sing": "div",
1931. "name": "div",
1932. "pos": 2,
1933. "type": ["bin"],
1934. "priority": [1],
1935. "calc": **function** (es) {
1936. **function** getExpressionValue(ex) {
1937. **if** (**typeof** (ex) == "number") {
1938. **return** ex;
1939. }
1940. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1941. **return** ex.object.attributes[ex.aIndex].value;
1942. }
1943. **if** (ex.expressions) {
1944. **return** ex.calculate();
1945. }
1946. }
1947. **if** (es.length == 2) {
1948. **return** parseInt(getExpressionValue(es[0]) / getExpressionValue(es[1]));
1949. }
1950. }
1951. }, {
1952. "sing": "mod",
1953. "name": "mod",
1954. "pos": 2,
1955. "type": ["bin"],
1956. "priority": [1],
1957. "calc": **function** (es) {
1958. **function** getExpressionValue(ex) {
1959. **if** (**typeof** (ex) == "number") {
1960. **return** ex;
1961. }
1962. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes && ex.object.attributes[ex.aIndex] && **typeof** (ex.object.attributes[ex.aIndex].value) == "number") {
1963. **return** ex.object.attributes[ex.aIndex].value;
1964. }
1965. **if** (ex.expressions) {
1966. **return** ex.calculate();
1967. }
1968. }
1969. **if** (es.length == 2) {
1970. **return** parseInt(getExpressionValue(es[0]) % getExpressionValue(es[1]));
1971. }
1972. }
1973. }]
1974. }
1976. AT\_KRL\_MathExpression.prototype.getThisSing = **function** () {
1977. **var** sgs = **this**.getAllSings();
1978. **for** (**var** i = 0; i < sgs.length; i++) {
1979. **if** (sgs[i].sing == **this**.sing) {
1980. **return** sgs[i];
1981. }
1982. }
1983. }
1985. AT\_KRL\_MathExpression.prototype.getKRL = **function** () {
1986. **var** sing = **this**.getThisSing();
1987. **var** res = '';
1989. **function** chKRL(ex) {
1990. **var** res = '';
1991. **if** (ex.object && ex.hasOwnProperty('aIndex') && ex.object.attributes[ex.aIndex] && AT\_KRL\_MathExpression.prototype.isNumber(ex)) {
1992. res += ex.object.name + '.' + ex.object.attributes[ex.aIndex].name;
1993. }
1994. **if** (ex.expressions && ex.getThisSing()) {
1995. res += ex.getKRL();
1996. }
1997. **if** (**typeof** (ex) == 'number') {
1998. res += ex.toString();
1999. }
2000. **return** res;
2001. }
2002. **if** (sing.type.indexOf('op') == -1) {
2003. **if** (sing.type.indexOf('coef') != -1 && **this**.expressions.length == 1) {
2004. res += '(' + **this**.sing + '(' + chKRL(**this**.expressions[0]) + '))';
2005. }
2006. **if** (sing.type.indexOf('bin') != -1 && **this**.expressions.length > 1) {
2007. res += '(';
2008. **for** (**var** i = 0; i < **this**.expressions.length; i++) {
2009. res += '(' + chKRL(**this**.expressions[i]) + ')' + ((i < **this**.expressions.length - 1) ? **this**.sing : '');
2010. }
2011. res += ')'
2012. }
2013. **if** (sing.type.length == 0) {
2014. res += chKRL(**this**.expressions[0]);
2015. }
2016. } **else** {
2017. **if** (sing.type.indexOf('bin') != -1) {
2018. res += **this**.sing + '(' + chKRL(**this**.expressions[0]) + ')(' + chKRL(**this**.expressions[1]) + ')';
2019. } **else** {
2020. res += **this**.sing + '(' + chKRL(**this**.expressions[0]) + ')';
2021. }
2022. }
2023. **return** res;
2024. }
2026. AT\_KRL\_MathExpression.prototype.calculate = **function** () {
2027. **var** sing = **this**.getThisSing();
2028. **return** sing.calc(**this**.expressions);
2029. }
2031. AT\_KRL\_MathExpression.prototype.hasObjAttrRef = **function** (name, index, deep) {
2032. **if** (**this**.expressions.length == 1 && !deep) {
2033. **if** (**this**.expressions[0].hasOwnProperty('object') && **this**.expressions[0].hasOwnProperty('aIndex')) {
2034. **return** (**this**.expressions[0].object.name == name && **this**.expressions[0].aIndex == index) && (**this**.sing == **null**);
2035. } **else**
2036. **if** (**this**.expressions[0].expressions) {
2037. **return** **this**.expressions[0].hasObjAttrRef(name, index, deep) && (**this**.sing == **null**);
2038. }
2039. }
2040. **if** (deep) {
2041. **var** res = **false**;
2042. **for** (**var** i = 0; i < **this**.expressions.length; i++) {
2043. **if** (**this**.expressions[i].hasOwnProperty('object') && **this**.expressions[0].hasOwnProperty('aIndex')) {
2044. res = res || (**this**.expressions[i].object.name == name && **this**.expressions[i].aIndex == index);
2045. } **else**
2046. **if** (**this**.expressions[i].expressions) {
2047. res = res || **this**.expressions[i].hasObjAttrRef(name, index, deep);
2048. }
2049. }
2050. **return** res;
2051. }
2052. **return** **false**;
2053. }
2055. AT\_KRL\_MathExpression.prototype.toXML = **function** () {
2056. **if** (**this**.sing == **null**) {
2057. **if** (**this**.expressions[0].object) {
2058. **var** r1 = **new** XMLDom('<ref />')
2059. r1.setAttribute('id', **this**.expressions[0].object.name);
2060. **var** r2 = **new** XMLDom('<ref />');
2061. r2.setAttribute('id', **this**.expressions[0].object.attributes[**this**.expressions[0].aIndex].name);
2062. r1.appendChild(r2);
2063. **return** r1.XML();
2064. } **else** {
2065. **return** **this**.expressions[0].toXML();
2066. }
2067. } **else** {
2068. **var** sg = **new** XMLDom('<math-' + **this**.getThisSing().name + ' />');
2069. **for** (**var** i = 0; i < **this**.expressions.length; i++) {
2070. sg.appendChild(**new** XMLDom(**this**.expressions[i].toXML()));
2071. }
2072. **return** sg.XML();
2073. }
2074. }
2075. AT\_KRL\_Fact.js
2076. **var** AT\_KRL\_Fact = **function** (leftside, rightside, sing, sure, exact) {
2078. **function** validateSides(l, r, s) {
2079. **var** relsForString = ['>=', '<=', '<', '>', '='];
2080. **var** relsForNumber = ['>=', '<=', '<', '>', '='];
2081. **var** numleft = AT\_KRL\_MathExpression.prototype.isNumber(l);
2082. **var** numrigth = AT\_KRL\_MathExpression.prototype.isNumber(r);
2084. **if** (numleft != numrigth) {
2085. **throw** **new** Error('Incompatible types for left and right sides to create a Fact');
2086. }
2087. **if** (numleft && relsForNumber.indexOf(s) == -1) {
2088. **throw** **new** Error('Bad relation type "' + s + '" for number types to create a Fact');
2089. }
2090. **if** (!numleft && relsForString.indexOf(s) == -1) {
2091. **throw** **new** Error('Bad relation type "' + s + '" for string or fuzzy types to create a Fact');
2092. }
2093. **return** [l, r, s];
2094. }
2096. **this**.leftside = validateSides(leftside, rightside, sing)[0];
2097. **this**.rightside = rightside;
2098. **this**.sing = sing;
2100. **function** validateShure(s) {
2101. **if** (s && s.length && s.length == 2 && s[0] >= 0 && s[0] <= s[1] && s[1] <= 100) {
2102. **return** s;
2103. } **else** {
2104. **return** **null**;
2105. }
2106. }
2108. **function** validateExact(e) {
2109. **if** (**typeof**(e) == "number" && 0 <= e && e <= 100) {
2110. **return** e;
2111. } **else** {
2112. **return** **null**;
2113. }
2114. }
2116. **this**.shure = [50, 100] || validateShure(shure);
2117. **this**.exact = parseFloat(validateExact(exact).toString() || '100');
2119. }
2121. AT\_KRL\_Fact.prototype.getKRL = **function** () {
2122. **var** res = '';
2123. res += **this**.leftside.getKRL();
2124. res += **this**.sing;
2125. res += **this**.rightside.getKRL();
2126. res += ' УВЕРЕННОСТЬ ' + JSON.stringify(**this**.shure) + ' ТОЧНОСТЬ ' + **this**.exact.toString();
2127. **return** res;
2128. }
2130. AT\_KRL\_Fact.prototype.getLeftSide = **function** () {
2131. **if** (AT\_KRL\_MathExpression.prototype.isNumber(**this**.leftside)) {
2132. **return** **this**.leftside.calculate();
2133. } **else** {
2134. **if** (**typeof**(**this**.leftside) == 'string') {
2135. **return** **this**.leftside.getKRL();
2136. } **else** {
2137. **return** **this**.leftside.expressions[0].object.attributes[**this**.leftside.expressions[0].aIndex].value.getKRL();
2138. }
2139. }
2140. }
2142. AT\_KRL\_Fact.prototype.getRightSide = **function** () {
2143. **if** (AT\_KRL\_MathExpression.prototype.isNumber(**this**.rightside)) {
2144. **return** **this**.rightside.calculate();
2145. } **else** {
2146. **if** (**typeof**(**this**.rightside) == 'string') {
2147. **return** **this**.rightside.getKRL();
2148. } **else** {
2149. **return** **this**.rightside.expressions[0].object.attributes[**this**.rightside.expressions[0].aIndex].value.getKRL();
2150. }
2151. }
2152. }
2154. AT\_KRL\_Fact.prototype.calculate = **function** () {
2155. **var** s = **this**.sing;
2156. s = (s.length == 1) ? s.replace('=', '==') : s;
2157. **var** l = **this**.getLeftSide();
2158. **var** r = **this**.getRightSide();
2159. **var** res = eval(l.toString() + s + r.toString());
2160. **return** res;
2162. }
2164. AT\_KRL\_Fact.prototype.hasObjAttrRef = **function**(name, index, deep){
2165. **if** (!deep && **this**.leftside.expressions){
2166. **return** (**this**.leftside.hasObjAttrRef(name,index,deep))
2167. }
2168. **if** (deep){
2169. **return** (**this**.leftside.hasObjAttrRef(name,index,deep) || **this**.rightside.hasObjAttrRef(name,index,deep))
2170. }
2171. }
2173. AT\_KRL\_Fact.prototype.toXML = **function**(){
2174. **var** xmlSing = {"=":"eq","<":"lt",">":"gt","<=":"le",">=":"ge"}[**this**.sing];
2175. **var** sg = **new** XMLDom('<' + xmlSing + ' />');
2176. **var** l = **new** XMLDom(**this**.leftside.toXML());
2177. **var** r = **new** XMLDom(**this**.rightside.toXML());
2178. sg.appendChild(l);
2179. sg.appendChild(r);
2180. **var** wh = **new** XMLDom('<with />')
2181. wh.setAttribute('belief',**this**.shure[0].toString());
2182. wh.setAttribute('probability',**this**.shure[1].toString());
2183. wh.setAttribute('accuracy',**this**.exact.toString());
2184. sg.appendChild(wh);
2185. **return** sg.XML();
2186. }
2187. AT\_KRL\_Rule.js
2188. **var** AT\_KRL\_ConditionNot = **function** (c) {
2189. **if** (c.getKRL) {
2190. **this**.condition = [c];
2191. } **else** {
2192. **throw** **new** Error('Invalid type of input condition or fact')
2193. }
2194. }
2196. AT\_KRL\_ConditionNot.prototype.getKRL = **function** () {
2197. **var** res = "~(" + **this**.condition[0].getKRL() + ")";
2198. **return** res;
2199. }
2201. AT\_KRL\_ConditionNot.prototype.calculate = **function** () {
2202. **return** !(**this**.condition[0].calculate());
2203. }
2205. AT\_KRL\_ConditionNot.prototype.toXML = **function**(){
2206. **var** not = **new** XMLDom('<not />')
2207. **for** (**var** i = 0; i < **this**.condition.length; i++){
2208. not.appendChild(**new** XMLDom(**this**.condition[i].toXML()));
2209. }
2210. **return** not.XML();
2211. }
2213. **var** AT\_KRL\_ConditionOr = **function** (cs) {
2214. **if** (cs && cs.length >= 2) {
2215. **this**.condition = cs;
2216. } **else** {
2217. **throw** **new** Error('Invalid type of input set of conditions or facts');
2218. }
2219. }
2221. AT\_KRL\_ConditionOr.prototype.getKRL = **function** () {
2222. **var** res = '(';
2223. **for** (**var** i = 0; i < **this**.condition.length; i++) {
2224. res += '(' + **this**.condition[i].getKRL() + ")|\n"
2225. }
2226. **return** res.slice(0, res.length - 2) + ')';
2227. }
2229. AT\_KRL\_ConditionOr.prototype.calculate = **function** () {
2230. **var** res = **false**;
2231. **for** (**var** i = 0; i < **this**.condition.length; i++) {
2232. res = res || **this**.condition[i].calculate();
2233. }
2234. **return** res;
2235. }
2237. AT\_KRL\_ConditionOr.prototype.toXML = **function**(){
2238. **var** or = **new** XMLDom('<or />')
2239. **for** (**var** i = 0; i < **this**.condition.length; i++){
2240. or.appendChild(**new** XMLDom(**this**.condition[i].toXML()));
2241. }
2242. **return** or.XML();
2243. }
2245. **var** AT\_KRL\_ConditionAnd = **function** (cs) {
2246. **if** (cs && cs.length >= 2) {
2247. **this**.condition = cs;
2248. } **else** {
2249. **throw** **new** Error('Invalid type of input set of conditions or facts');
2250. }
2251. }
2253. AT\_KRL\_ConditionAnd.prototype.getKRL = **function** () {
2254. **var** res = '(';
2255. **for** (**var** i = 0; i < **this**.condition.length; i++) {
2256. res += '(' + **this**.condition[i].getKRL() + ")&\n"
2257. }
2258. **return** res.slice(0, res.length - 2) + ')';
2259. }
2261. AT\_KRL\_ConditionAnd.prototype.calculate = **function** () {
2262. **var** res = **true**;
2263. **for** (**var** i = 0; i < **this**.condition.length; i++) {
2264. res = res && **this**.condition[i].calculate();
2265. }
2266. **return** res;
2267. }
2269. AT\_KRL\_ConditionAnd.prototype.toXML = **function**(){
2270. **var** and = **new** XMLDom('<and />')
2271. **for** (**var** i = 0; i < **this**.condition.length; i++){
2272. and.appendChild(**new** XMLDom(**this**.condition[i].toXML()));
2273. }
2274. **return** and.XML();
2275. }
2277. **var** AT\_KRL\_Rule = **function** (a, cs, fs, n, e) {
2278. **var** self = **this**;
2280. **function** validateResult(fs) {
2281. **var** r = fs.result ? (fs.result.length ? fs.result : [fs.result]) : (fs.length ? fs : [fs]);
2282. **var** res = **true**;
2283. **for** (**var** i = 0; i < r.length; i++) {
2284. **if** (!self.factIsSimple(r[i])) {
2285. **throw** **new** Error('Invalid format of conclusion facts to create Rule');
2286. }
2287. res = res && self.factIsSimple(r[i]);
2288. }
2289. **if** (fs.elsresult) {
2290. r = fs.elsresult.length ? fs.elsresult : [fs.elsresult];
2291. **for** (**var** i = 0; i < r.length; i++) {
2292. **for** (**var** i = 0; i < r.length; i++) {
2293. **if** (!self.factIsSimple(r[i])) {
2294. **throw** **new** Error('Invalid format of conclusion facts to create Rule');
2295. }
2296. res = res && self.factIsSimple(r[i]);
2297. }
2298. }
2299. }
2300. **return** res;
2301. }
2302. **var** verificated = **true** && **this**.conditionHasTwoSides(cs) && validateResult(fs);
2304. **if** (verificated) {
2305. **function** validateName(s) {
2306. **var** inv = '~!@#$%^&\*()-=+/\*|\\?.,:;`\'" ';
2307. **for** (**var** i = 0; i < inv.length; i++) {
2308. **if** (s.indexOf(inv[i]) != -1) {
2309. **throw** **new** Error('Invalid character "' + inv[i] + '" in name ' + s);
2310. }
2311. }
2312. **return** s;
2313. }
2314. **this**.name = validateName(a.replaceAll(' ', '') || ("ПРАВИЛО" + e.rules.length));
2315. **this**.comment = n || **this**.name;
2316. **this**.ifConds = cs;
2317. **this**.result = fs.result ? (fs.result.length ? fs.result : [fs.result]) : (fs.length ? fs : [fs]);
2318. **this**.elsresult = fs.elsresult ? (fs.elsresult.length ? fs.elsresult : [fs.elsresult]) : [];
2319. e.pushRule(**this**);
2320. **this**.editor = e;
2321. } **else** {
2322. **throw** **new** Error('Wrong format of parameters to create rule, condition must have two expressions in minimum');
2323. }
2324. }
2326. AT\_KRL\_Rule.prototype.conditionHasTwoSides = **function** (cs) {
2327. **if** (cs.condition.length > 1) {
2328. **return** **true**;
2329. }
2330. **if** (cs.condition[0].condition) {
2331. **return** **this**.conditionHasTwoSides(cs.condition[0]);
2332. }
2333. **return** **false**;
2334. }
2336. AT\_KRL\_Rule.prototype.sideHasNoObject = **function** (side) {
2337. **if** (**typeof** (side) == "number" || **typeof** (side) == "string") {
2338. **return** **true**;
2339. }
2340. **if** (side.expressions) {
2341. **var** res = **true**;
2342. **for** (**var** i = 0; i < side.expressions.length; i++) {
2343. res = res && **this**.sideHasNoObject(side.expressions[i]);
2344. }
2345. **return** res;
2346. }
2347. **return** **false**;
2348. }
2350. AT\_KRL\_Rule.prototype.sideIsOnlyObject = **function** (side) {
2351. **if** (side.object && side.aIndex != **null** && side.object.attributes[side.aIndex]) {
2352. **return** **true**;
2353. }
2354. **if** (side.expressions && side.expressions.length == 1) {
2355. **return** **this**.sideIsOnlyObject(side.expressions[0]);
2356. }
2357. **return** **false**;
2358. }
2360. AT\_KRL\_Rule.prototype.factIsSimple = **function** (f, l) {
2361. **var** res = **this**.sideIsOnlyObject(f.leftside) && **this**.sideHasNoObject(f.rightside);
2362. **if** (!res) {
2363. **throw** **new** SyntaxError('Описание факта "' + l + '" не подходит для описания результата правила\nТребуемый формат: ОБЪЕКТ.АТРИБУТ=значение');
2364. }
2365. **return** res;
2366. }
2368. AT\_KRL\_Rule.prototype.getKRL = **function** () {
2369. **var** res = "ПРАВИЛО " + **this**.name + "\nЕСЛИ\n " + **this**.ifConds.getKRL() + '\nТО\n';
2370. **for** (**var** i = 0; i < **this**.result.length; i++) {
2371. res += **this**.result[i].getKRL() + '\n';
2372. }
2373. **if** (**this**.elsresult && **this**.elsresult.length > 0) {
2374. res += 'ИНАЧЕ\n'
2375. **for** (**var** i = 0; i < **this**.elsresult.length; i++) {
2376. res += **this**.elsresult[i].getKRL() + '\n';
2377. }
2378. }
2379. res += "КОММЕНТАРИЙ " + **this**.comment + '\n\n';
2380. **return** res;
2381. };
2383. AT\_KRL\_Rule.prototype.hasObjAttrRef = **function** (name, index, deep) {
2384. **if** (!deep) {
2385. **var** res = **false**;
2386. **var** facts = **this**.result.concat(**this**.elsresult);
2387. **for** (**var** i = 0; i < facts.length; i++) {
2388. res = res || facts[i].hasObjAttrRef(name, index, deep);
2389. }
2390. **return** res;
2391. } **else** {
2392. **var** KRL = **this**.getKRL();
2393. **var** obj;
2394. **for** (**var** i = 0; i < **this**.editor.objects.length; i++) {
2395. **var** o = **this**.editor.objects[i];
2396. **if** (name == o.name) {
2397. obj = o;
2398. **break**;
2399. }
2400. }
2401. **var** aName = obj.attributes[index].name;
2402. **return** (KRL.indexOf(name + '.' + aName) != -1);
2403. }
2404. }
2406. AT\_KRL\_Rule.prototype.toXML = **function** (index) {
2407. **var** num = index || 0;
2408. **var** rule = **new** XMLDom('<rule />');
2409. rule.setAttribute('id',(num+1).toString());
2410. rule.setAttribute('meta','simple');
2411. rule.setAttribute('desc',**this**.comment);
2412. **var** condition = **new** XMLDom('<condition />');
2413. condition.appendChild(**new** XMLDom(**this**.ifConds.toXML()));
2414. rule.appendChild(condition);
2416. **var** action = **new** XMLDom('<action />');
2417. **for** (**var** i = 0; i < **this**.result.length; i++) {
2418. **var** assign = **new** XMLDom('<assign />');
2419. **var** eq = **new** XMLDom(**this**.result[i].toXML());
2420. assign.appendChild(eq.getChildNodes()[0]);
2421. assign.appendChild(eq.getChildNodes()[1]);
2422. action.appendChild(assign);
2423. }
2424. rule.appendChild(action);
2426. **if** (**this**.elsresult.length != 0) {
2427. **var** elseAction = **new** XMLDom('<else-action />');
2428. **for** (**var** i = 0; i < **this**.elsresult.length; i++) {
2429. **var** assign = **new** XMLDom('<assign />');
2430. **var** eq = **new** XMLDom(**this**.elsresult[i].toXML());
2431. assign.appendChild(eq.getChildNodes()[0]);
2432. assign.appendChild(eq.getChildNodes()[1]);
2433. elseAction.appendChild(assign);
2434. }
2435. rule.appendChild(elseAction);
2436. }
2437. **return** rule.XML();
2438. }
2439. AT\_KRL\_Parser.js
2440. **var** AT\_KRL\_Parser = **function** (e) {
2441. **this**.editor = e || **new** AT\_KRL\_Editor();
2442. }
2444. //------Предобработка текста------
2445. AT\_KRL\_Parser.prototype.getAllDeclarations = **function** (krl) { //Улучшить, чтобы не обязательно был бы \n между каждым объявлением
2446. **var** text = (krl || **this**.editor.getKRL()) + "\n";
2447. **var** ds = [];
2448. text = text.replaceAll('\n\n\n', '\n\n');
2449. **while** (text != "") {
2450. **var** i = text.indexOf('\n\n') != -1 ? text.indexOf('\n\n') : (text.length - 1);
2451. **var** d = text.slice(0, i);
2452. **if** (d != "") {
2453. ds.push(d);
2454. }
2455. i = (i + 2 < text.length) ? i : (i - 2);
2456. text = text.substr(i + 2, text.length - 1);
2457. }
2458. **return** ds;
2459. }
2461. AT\_KRL\_Parser.prototype.getAllLines = **function** (declaration) {
2462. **var** lines = [];
2463. **var** d = declaration + "\n";
2464. **while** (d != "") {
2465. **var** i = d.indexOf("\n");
2466. **var** line = d.substring(0, i);
2467. **if** (line != "") {
2468. lines.push(line);
2469. }
2470. **if** (i = d.length - 1) {
2471. i--;
2472. }
2473. d = d.substring(d.indexOf("\n") + 1, d.length);
2474. }
2475. **return** lines;
2476. }
2478. AT\_KRL\_Parser.prototype.getTextFromLines = **function** (lines) {
2479. **var** res = "";
2480. **for** (**var** i = 0; i < lines.length; i++) {
2481. res += lines[i] + '\n';
2482. }
2483. **return** res;
2484. }
2486. //------Вспомогательное------
2487. AT\_KRL\_Parser.prototype.getAllTypes = **function** () {
2488. **return** (**new** AT\_KRL\_Editor()).getAllTypes();
2489. }
2491. AT\_KRL\_Parser.prototype.validateName = **function** (n, pl) {
2492. **var** invalidSymbols = "'\"`~!@#$%^&\*()+-=/\*{[]}\\|,<.>/? ";
2493. **for** (**var** i = 0; i < invalidSymbols.length; i++) {
2494. **if** (n.indexOf(invalidSymbols[i]) != -1) {
2495. **throw** **new** SyntaxError("Недопустимый символ: " + invalidSymbols[i].replace(" ", "ПРОБЕЛ") + " В имени: " + n + " На позиции" + n.indexOf(invalidSymbols[i]));
2496. }
2497. }
2498. **var** names = [];
2499. **for** (**var** i = 0; i < **this**.editor[pl].length; i++) {
2500. names.push(**this**.editor[pl][i].name);
2501. }
2502. **if** (names.indexOf(n) != -1) {
2503. **throw** **new** SyntaxError('Имя "' + n + '" уже сущетсвует');
2504. }
2505. **return** n;
2506. }
2508. //------Парсинг типа------
2509. AT\_KRL\_Parser.prototype.parseTypeName = **function** (line) {
2510. **if** (line.indexOf("ТИП ") != 0) {
2511. **throw** **new** SyntaxError("Невозможно получить имя типа из: " + line);
2512. }
2513. **var** name = **this**.validateName(line.replace('ТИП ', ''), 'types');
2514. **return** name;
2515. }
2517. AT\_KRL\_Parser.prototype.parseTypeNameFL = **function** (lines) {
2518. **return** **this**.parseTypeName(lines[0]);
2519. }
2521. AT\_KRL\_Parser.prototype.parseTypeParentFL = **function** (lines) {
2522. **var** res = NaN;
2523. **var** err = "";
2524. **for** (**var** i = 0; i < lines.length; i++) {
2525. **var** l = lines[i];
2526. **switch** (l) {
2527. **case** "ЧИСЛО":
2528. res = 0;
2529. **break**;
2530. **case** "СИМВОЛ":
2531. res = 1;
2532. **break**;
2533. **case** "НЕЧЕТКИЙ":
2534. res = 2;
2535. **break**;
2536. **default**:
2537. **break**;
2538. }
2539. err += l + "\n";
2540. }
2541. **if** (isNaN(res)) {
2542. **throw** **new** SyntaxError('Невозможно получить изначальный тип ("ЧИСЛО", "СИМВОЛ" или "НЕЧЕТКИЙ") из:\n' + err);
2543. }
2544. **return** res;
2545. }
2547. AT\_KRL\_Parser.prototype.parseCoordinates = **function** (l) {
2548. **var** coordinates = [];
2549. **if** (l.charAt(0) != '{' || l.charAt(l.length - 1) != '}') {
2550. **throw** **new** SyntaxError('Неверный синтаксис задания координат функции принадлежности в объявлении: ' + l);
2551. }
2553. **function** sNum(c, s) {
2554. **var** res = 0;
2555. **for** (**var** i = 0; i < s.length; i++) {
2556. **if** (c == s.slice(i, i + c.length)) {
2557. res++;
2558. }
2559. }
2560. **return** res;
2561. }
2562. **if** (sNum('|', l) != sNum(';', l) + 1) {
2563. **throw** **new** SyntaxError('Неверный синтаксис задания координат функции принадлежности в объявлении: ' + l);
2564. }
2565. **var** state = 'ini';
2566. **var** i = 0;
2567. **var** x = '';
2568. **var** y = '';
2569. **while** (i < l.length) {
2571. **if** (state == 'parseX' && l.charAt(i) == '|') {
2572. state = "parseY";
2573. i++;
2574. }
2575. **if** (state == 'parseY' && (l.charAt(i) == ';' || l.charAt(i) == '}')) {
2576. state = 'makePair';
2577. }
2579. **if** (state == 'makePair' || state == "ini") {
2580. **if** (state != "ini") {
2581. **if** (x == '' || y == '') {
2582. **throw** **new** SyntaxError('Некорректный символ в определении координат функции принадлежности \nВ объявлении ' + l + '\nНа позиции ' + i);
2583. }
2585. **var** cs = {
2586. "X": parseFloat(x),
2587. "Y": parseFloat(y)
2588. };
2589. coordinates.push(cs);
2590. }
2591. x = '';
2592. y = '';
2593. **while** (isNaN(parseInt(l.charAt(i))) && i < l.length) {
2594. i++;
2595. }
2596. state = 'parseX';
2597. **if** (i == l.length) {
2598. state = 'end';
2599. }
2600. }
2602. **if** (state == 'parseX') {
2603. **if** (isNaN(parseInt(l.charAt(i))) && l.charAt(i) != ',' && l.charAt(i) != '.') {
2604. **throw** **new** SyntaxError('Некорректный символ в определении координат функции принадлежности \nВ объявлении ' + l + '\nНа позиции ' + i);
2605. }
2606. x += l.charAt(i).replace(',', '.');
2607. }
2609. **if** (state == 'parseY') {
2610. **if** (isNaN(parseInt(l.charAt(i))) && l.charAt(i) != ',' && l.charAt(i) != '.') {
2611. **throw** **new** SyntaxError('Некорректный символ в определении координат функции принадлежности \nВ объявлении ' + l + '\nНа позиции ' + i);
2612. }
2613. y += l.charAt(i).replace(',', '.');
2614. }
2615. i++;
2616. }
2617. **return** coordinates;
2618. }
2620. AT\_KRL\_Parser.prototype.parseFP = **function** (line, v) {
2621. **if** (!line || line.indexOf("КОММЕНТАРИЙ ") == 0) {
2622. **throw** **new** SyntaxError("Не совпадают количество объявлений функций принадлежности и количество значений типа");
2623. }
2624. **if** (line.slice(0, 1) != '"' || line.lastIndexOf('" ') == -1) {
2625. **throw** **new** SyntaxError("Невозможно получить имя функции принадлежности в объявлении: " + line);
2626. }
2627. **var** name = line.substring(1, line.lastIndexOf('" '));
2628. **if** (v.indexOf(name) == -1) {
2629. **throw** **new** SyntaxError('Имя функции принадлежности должно быть равно одному из символьных щначений, указанном в типе (' + JSON.stringify(v).slice(0, JSON.stringify(v).length - 1).slice(1) + ')');
2630. }
2631. **var** l = line.slice(name.length + 3);
2632. **var** minS = l.slice(0, l.indexOf(' '));
2633. **if** (minS == '' || isNaN(parseFloat(minS))) {
2634. **throw** **new** SyntaxError('Невозможно получить минимум абсциссы функции принадлежности в объявлении: ' + line);
2635. }
2636. **var** min = parseFloat(minS);
2637. l = l.slice(minS.length + 1);
2638. **var** maxS = l.slice(0, l.indexOf(' '));
2639. **if** (maxS == '' || isNaN(parseFloat(maxS))) {
2640. **throw** **new** SyntaxError('Невозможно получить максимум абсциссы функции принадлежности в объявлении: ' + line);
2641. }
2642. **var** max = parseFloat(maxS);
2643. **if** (max <= min) {
2644. **throw** **new** SyntaxError('Минимум абсциссы функции принадлежности не должен быть больше или равен максимуму абсциссы в объявлении: ' + line);
2645. }
2646. l = l.slice(maxS.length + 1);
2647. **var** lgt = l.slice(0, l.indexOf(' '));
2648. l = l.slice(l.indexOf('={') + 1);
2649. **if** (l.indexOf('{') != 0) {
2650. **throw** **new** SyntaxError('Неверный синтаксис задания координат функции принадлежности в объявлении: ' + line);
2651. }
2652. **var** coordinates = **this**.parseCoordinates(l);
2653. **if** (coordinates.length != parseInt(lgt)) {
2654. **throw** **new** SyntaxError("Несовпадение заданного числа и количества координат в объявлении: " + line);
2655. }
2656. **var** FP = {
2657. "value": name,
2658. "min": min,
2659. "max": max,
2660. "coordinates": coordinates
2661. };
2662. **return** FP;
2663. }
2665. AT\_KRL\_Parser.prototype.parseTypeParamsFL = **function** (lines, p) {
2666. **var** res = {
2667. "values": []
2668. };
2669. **switch** (p) {
2670. **case** 0:
2671. **var** v = [parseFloat(lines[lines.indexOf("ЧИСЛО") + 1].replace('ОТ ', '').replaceAll(',', '.')), parseFloat(lines[lines.indexOf("ЧИСЛО") + 2].replace('ДО ', '').replaceAll(',', '.'))];
2672. **if** (isNaN(v[0]) || isNaN(v[1])) {
2673. **throw** **new** SyntaxError('Несовпадение типа и значений в объявлении:\n' + **this**.getTextFromLines(lines));
2674. };
2675. res.values = v;
2676. **return** res;
2677. **case** 1:
2678. **var** v = [];
2679. **for** (**var** i = lines.indexOf("СИМВОЛ") + 1; i < lines.length; i++) {
2680. **if** (lines[i].indexOf('КОММЕНТАРИЙ ') != 0) {
2681. **if** (lines[i].slice(0, 1) != '"' || lines[i].slice(lines[i].length - 1) != '"') {
2682. **throw** **new** SyntaxError('Неверный синтаксис задания символьного значения в объявлении:\n' + **this**.getTextFromLines(lines) + '\nСтрока: ' + i);
2683. }
2684. v.push(lines[i].slice(1, lines[i].length - 1));
2685. }
2686. }
2687. res.values = v;
2688. **return** res;
2689. **break**;
2690. **case** 2:
2691. **var** v = [];
2692. **var** i = lines.indexOf("СИМВОЛ") + 1;
2693. **var** j = lines.indexOf("НЕЧЕТКИЙ");
2694. **if** (j == -1 || i == 0) {
2695. **throw** **new** SyntaxError('Неверный синтаксис задания нечеткого типа в объявлении:\n' + **this**.getTextFromLines(lines) + '\nСтрока: ' + 0);
2696. }
2697. **while** (i < j) {
2698. **if** (lines[i].slice(0, 1) != '"' || lines[i].slice(lines[i].length - 1) != '"') {
2699. **throw** **new** SyntaxError('Неверный синтаксис задания символьного значения в объявлении:\n' + **this**.getTextFromLines(lines) + '\nСтрока: ' + i);
2700. }
2701. v.push(lines[i].slice(1, lines[i].length - 1));
2702. i++;
2703. }
2704. **if** (parseInt(lines[j + 1]) != v.length) {
2705. **throw** **new** SyntaxError('Несовпадение количества значений типа и количества функций принадлежности в объявлении:\n' + **this**.getTextFromLines(lines) + '\nСтрока: ' + (j + 1));
2706. }
2707. **var** pos = j + 2;
2708. **var** FP = [];
2709. **for** (**var** i = 0; i < v.length; i++) {
2710. FP.push(**this**.parseFP(lines[pos + i], v));
2711. }
2712. res.values = v;
2713. res.FP = FP;
2714. **return** res;
2715. **break**;
2716. }
2717. }
2719. AT\_KRL\_Parser.prototype.parseCommentFL = **function** (lines) {
2720. **if** (lines[lines.length - 1].indexOf('КОММЕНТАРИЙ ') == 0) {
2721. **return** lines[lines.length - 1].replace('КОММЕНТАРИЙ ', '');
2722. } **else** {
2723. **return** 'Пустой комментарий';
2724. }
2725. }
2727. AT\_KRL\_Parser.prototype.parseType = **function** (declaration) {
2728. **var** lines = **this**.getAllLines(declaration);
2729. **var** name = **this**.parseTypeNameFL(lines);
2730. **var** parent = **this**.parseTypeParentFL(lines);
2731. **var** params = **this**.parseTypeParamsFL(lines, parent);
2732. **var** comment = **this**.parseCommentFL(lines);
2733. **var** type = **new** AT\_KRL\_Type(name, parent, params, comment, **this**.editor);
2734. **return** type;
2735. }
2737. //------Парсинг объекта------
2738. AT\_KRL\_Parser.prototype.parseObjectName = **function** (line) {
2739. **if** (line.indexOf("ОБЪЕКТ ") != 0) {
2740. **throw** **new** SyntaxError("Невозможно получить имя объекта из: " + line);
2741. }
2742. **var** name = **this**.validateName(line.replace('ОБЪЕКТ ', ''), 'objects');
2743. **return** name;
2744. }
2746. AT\_KRL\_Parser.prototype.parseObjectNameFL = **function** (lines) {
2747. **return** **this**.parseObjectName(lines[0]);
2748. }
2750. AT\_KRL\_Parser.prototype.parseAttributeName = **function** (line) {
2751. **if** (line.indexOf("АТРИБУТ ") != 0) {
2752. **throw** **new** SyntaxError("Невозможно получить имя атрибута из: " + line);
2753. }
2754. **var** name = line.replace('АТРИБУТ ', '');
2755. **return** name;
2756. }
2758. AT\_KRL\_Parser.prototype.parseAttributeNameFL = **function** (lines) {
2759. **return** **this**.parseAttributeName(lines[0]);
2760. }
2762. AT\_KRL\_Parser.prototype.parseAttributeTypeFL = **function** (lines) {
2763. **if** (lines[1].indexOf('ТИП ') != 0) {
2764. **throw** **new** SyntaxError('Невозможно получить имя типа для атрибута в объявлении:\n' + **this**.getTextFromLines(lines) + '\nСтрока 1');
2765. }
2766. **var** name = lines[1].replace('ТИП ', '');
2767. **var** type = **false**;
2768. **for** (**var** i = 0; i < **this**.editor.types.length; i++) {
2769. **if** (**this**.editor.types[i].name == name) {
2770. type = **this**.editor.types[i];
2771. }
2772. }
2773. **if** (!type) {
2774. **throw** **new** SyntaxError('Атрибут имеет неизвестный тип в объявлении:\n' + **this**.getTextFromLines(lines) + '\nСтрока 1');
2775. }
2776. **return** type;
2777. }
2779. AT\_KRL\_Parser.prototype.parseObjectAttribute = **function** (ls, i) {
2780. **var** lines = [ls[i], ls[i + 1], ls[i + 2]];
2781. **var** name = **this**.parseAttributeNameFL(lines);
2782. **var** type = **this**.parseAttributeTypeFL(lines);
2783. **var** comment = **this**.parseCommentFL(lines);
2784. **var** attribute = **new** AT\_KRL\_Attribute(name, type, type.values[0], comment);
2785. **return** attribute;
2786. }
2788. AT\_KRL\_Parser.prototype.validateAttribute = **function** (attribute, attributes) {
2789. **var** names = [];
2790. **for** (**var** i = 0; i < attributes.length; i++) {
2791. names.push(attributes[i].name);
2792. }
2793. **if** (names.indexOf(attribute.name) != -1) {
2794. **throw** **new** SyntaxError('Повторное объявление атрибута ' + name);
2795. }
2796. **return** attribute;
2797. }
2799. AT\_KRL\_Parser.prototype.parseObjectAttributesFl = **function** (lines) {
2800. **var** attributes = [];
2801. **for** (**var** i = 0; i < lines.length; i++) {
2802. **if** (lines[i].indexOf('АТРИБУТ ') == 0) {
2803. attributes.push(**this**.validateAttribute(**this**.parseObjectAttribute(lines, i), attributes))
2804. }
2805. }
2806. **return** attributes;
2807. }
2809. AT\_KRL\_Parser.prototype.parseObject = **function** (declaration) {
2810. **var** lines = **this**.getAllLines(declaration);
2811. **var** name = **this**.parseObjectNameFL(lines);
2812. **var** attributes = **this**.parseObjectAttributesFl(lines);
2813. **var** comment = **this**.parseCommentFL(lines)
2814. **var** object = **new** AT\_KRL\_Object(name, attributes, comment, **this**.editor);
2815. **return** object;
2816. }
2818. //------Парсинг математических методом построения и обхода дерева разбора и парсинг строковых или нечетких выражений------//
2819. AT\_KRL\_Parser.prototype.parseStFzExpression = **function** (subline) {
2820. **if** (subline.indexOf('"') == 0 && subline.lastIndexOf('"') == subline.length - 1) {
2821. **return** subline.substr(1, subline.length - 2);
2822. } **else** {
2823. **var** StFz = subline;
2824. **var** obj = StFz.substring(0, StFz.lastIndexOf('.'));
2825. **var** att = StFz.substring(StFz.lastIndexOf('.') + 1, StFz.length);
2827. **var** object = **null**;
2829. **for** (**var** i = 0; i < **this**.editor.objects.length; i++) {
2830. **if** (**this**.editor.objects[i].name == obj) {
2831. object = **this**.editor.objects[i];
2832. }
2833. }
2834. **if** (!object) {
2835. **throw** **new** SyntaxError('Недопустимое имя объекта "' + obj + '" в объявлении:\n' + line);
2836. }
2837. **var** aIndex = -1;
2838. **for** (**var** i = 0; i < object.attributes.length; i++) {
2839. **if** (att == object.attributes[i].name) {
2840. aIndex = i;
2841. }
2842. }
2843. **if** (aIndex == -1) {
2844. **throw** **new** SyntaxError('Недопустимое имя атрибута "' + att + '" в объявлении:\n' + line);
2845. }
2846. **if** (object.attributes[aIndex].type.vType != 1 && object.attributes[aIndex].type.vType != 2) {
2847. **throw** **new** SyntaxError('Атрибут ' + obj + '.' + att + ' имеет недопустимый тип\nОжидался символьный или нечеткий тип атрибута');
2848. }
2849. **return** **new** AT\_KRL\_StFzExpression(object, aIndex);
2850. }
2851. }
2853. AT\_KRL\_Parser.prototype.parseSimpleMathExpression = **function** (subline) {
2854. **var** obj = subline.substring(0, subline.lastIndexOf('.'));
2855. **var** att = subline.substring(subline.lastIndexOf('.') + 1, subline.length);
2857. **var** object = **null**;
2859. **for** (**var** i = 0; i < **this**.editor.objects.length; i++) {
2860. **if** (**this**.editor.objects[i].name == obj) {
2861. object = **this**.editor.objects[i];
2862. }
2863. }
2864. **if** (!object) {
2865. **throw** **new** SyntaxError('Недопустимое имя объекта "' + obj + '" в объявлении:\n' + subline);
2866. }
2867. **var** aIndex = -1;
2868. **for** (**var** i = 0; i < object.attributes.length; i++) {
2869. **if** (att == object.attributes[i].name) {
2870. aIndex = i;
2871. }
2872. }
2873. **if** (aIndex == -1) {
2874. **throw** **new** SyntaxError('Недопустимое имя атрибута "' + att + '" в объявлении:\n' + subline);
2875. }
2876. **if** (object.attributes[aIndex].type.vType != 0) {
2877. **throw** **new** SyntaxError('Атрибут ' + obj + '.' + att + ' имеет недопустимый тип\nОжидался числовой тип атрибута');
2878. }
2879. **var** m = **new** AT\_KRL\_MathExpression({
2880. object: object,
2881. aIndex: aIndex
2882. });
2883. **return** m;
2884. }
2886. AT\_KRL\_Parser.prototype.getToNextMathOperation = **function** (subline, sNames, index, current) {
2887. **var** i = index + current.length;
2888. **var** bracket = (subline.charAt(i) == '(') ? 1 : 0;
2890. **function** isOperation(s, n, i) {
2891. **var** res = **false**;
2892. **for** (**var** k = 0; k < n.length; k++) {
2893. res = res || ((s.indexOf(n[k], i) == i) && (n.indexOf(s.substr(i, n[k].length + 1)) == -1));
2894. **if** (res) {
2895. **return** n[k];
2896. }
2897. }
2898. **return** **false**;
2899. }
2900. **var** o = isOperation(subline, sNames, i);
2901. **while** ((!isOperation(subline, sNames, i) || isOperation(subline, sNames, i) && bracket != 0) && i < subline.length - 1) {
2902. i++;
2903. o = isOperation(subline, sNames, i);
2904. **if** (subline.charAt(i) == "(") {
2905. bracket++;
2906. }
2907. **if** (subline.charAt(i) == ")") {
2908. bracket--;
2909. }
2910. }
2911. **if** (i == subline.length - 1) {
2912. **return** **false**;
2913. }
2914. **return** {
2915. index: i,
2916. operation: o
2917. };
2918. }
2920. AT\_KRL\_Parser.prototype.getToFirstMathOperation = **function** (subline, sNames) {
2921. **return** **this**.getToNextMathOperation(subline, sNames, 0, '');
2922. }
2924. AT\_KRL\_Parser.prototype.getLineFromBrackets = **function** (line, index) {
2925. **var** i = index;
2926. **var** res = '';
2927. **while** (line.charAt(i) != '(' && i < line.length) {
2928. i++;
2929. }
2930. **if** (line.charAt(i) == '(') {
2931. **var** b = 1;
2932. **while** (i < line.length && b != 0) {
2933. i++;
2934. **if** (line.charAt(i) == '(') {
2935. b++;
2936. }
2937. **if** (line.charAt(i) == ')') {
2938. b--;
2939. }
2940. **if** (b != 0) {
2941. res += line.charAt(i)
2942. }
2943. }
2944. **return** {
2945. res: res,
2946. index: i
2947. };
2948. }
2949. **return** {
2950. res: '',
2951. index: -1
2952. };
2953. }
2955. AT\_KRL\_Parser.prototype.getLineFromFirstBrackets = **function** (line) {
2956. **return** **this**.getLineFromBrackets(line, 0);
2957. }
2959. AT\_KRL\_Parser.prototype.getOperationSides = **function** (subline, operation, index) {
2960. **function** isInBrackets(line) {
2961. **if** (line.charAt(0) != '(') {
2962. **return** **false**;
2963. } **else** {
2964. **var** b = 1;
2965. **var** i = 1;
2966. **var** index = 0;
2967. **while** (b != 0 && i < line.length) {
2968. **if** (line.charAt(i) == '(') {
2969. b++;
2970. }
2971. **if** (line.charAt(i) == ')') {
2972. b--;
2973. }
2974. **if** (b == 0 && index == 0) {
2975. index = i;
2976. }
2977. **if** (b == 0 && i != line.length - 1) {
2978. **return** **false**;
2979. }
2980. i++;
2981. }
2982. **if** (b == 0 && index == line.length - 1) {
2983. **return** **true**;
2984. }
2985. }
2986. }
2988. **var** left;
2989. **var** right;
2990. **if** (operation.sing != 'log') {
2991. left = subline.substring(0, index);
2992. right = subline.substring(index + operation.sing.length, subline.length);
2993. } **else** {
2994. left = **this**.getLineFromBrackets(subline, index + 3).res;
2995. right = **this**.getLineFromBrackets(subline, index + left.length + 4).res;
2996. }
2997. **while** (isInBrackets(left)) {
2998. left = **this**.getLineFromFirstBrackets(left).res;
2999. }
3000. **while** (isInBrackets(right)) {
3001. right = **this**.getLineFromFirstBrackets(right).res;
3002. }
3003. **if** (left == '') {
3004. **return** [right];
3005. }
3006. /\***if** (){
3007. **throw** **new** SyntaxError('Невозможно интерпретировать математическое выражение в объявлении:\n' + subline);
3008. }\*/
3009. **return** [left, right];
3010. }
3012. AT\_KRL\_Parser.prototype.getMathExpressionTree = **function** (subline) {
3013. **var** inv = '`~!@"№$%&<>?=';
3015. **function** isInBrackets(line) {
3016. **if** (line.charAt(0) != '(') {
3017. **return** **false**;
3018. } **else** {
3019. **var** b = 1;
3020. **var** i = 1;
3021. **var** index = 0;
3022. **while** (b != 0 && i < line.length) {
3023. **if** (line.charAt(i) == '(') {
3024. b++;
3025. }
3026. **if** (line.charAt(i) == ')') {
3027. b--;
3028. }
3029. **if** (b == 0 && index == 0) {
3030. index = i;
3031. }
3032. **if** (b == 0 && i != line.length - 1) {
3033. **return** **false**;
3034. }
3035. i++;
3036. }
3037. **if** (b == 0 && index == line.length - 1) {
3038. **return** **true**;
3039. }
3040. }
3041. }
3042. **while** (isInBrackets(subline)) {
3043. subline = **this**.getLineFromFirstBrackets(subline).res;
3044. }
3045. **for** (**var** i = 0; i < inv.length; i++) {
3046. **if** (subline.indexOf(inv[i]) != -1) {
3047. **throw** **new** SyntaxError('Недопустимый символ "' + inv[i] + '" в объявлении ' + subline + ' на позиции ' + subline.indexOf(inv[i]));
3048. }
3049. }
3050. **var** sings = AT\_KRL\_MathExpression.prototype.getAllSings();
3051. **var** sNames = [];
3052. **for** (**var** i = 0; i < sings.length; i++) {
3053. **if** (sings[i].sing) {
3054. sNames.push(sings[i].sing);
3055. }
3056. }
3058. **function** getOperationByName(name) {
3059. **var** s = AT\_KRL\_MathExpression.prototype.getAllSings();
3060. **for** (**var** i = 0; i < s.length; i++) {
3061. **if** (s[i].sing == name) {
3062. **return** s[i];
3063. }
3064. }
3065. **return** **null**;
3066. }
3068. **var** thisLayerOperations = [];
3069. **var** o = **this**.getToFirstMathOperation(subline, sNames);
3070. **while** (o) {
3071. thisLayerOperations.push({
3072. "operation": getOperationByName(o.operation),
3073. "index": o.index
3074. });
3075. o = **this**.getToNextMathOperation(subline, sNames, o.index, o.operation);
3076. }
3078. **if** (thisLayerOperations.length != 0) {
3079. **var** o = thisLayerOperations[0];
3080. **for** (**var** i = 1; i < thisLayerOperations.length; i++) {
3081. **if** (thisLayerOperations[i].operation.priority[0] <= o.operation.priority[0]) {
3082. o = thisLayerOperations[i];
3083. }
3084. }
3085. **var** sides = **this**.getOperationSides(subline, o.operation, o.index);
3086. **if** (sides.length == 1) {
3087. **return** {
3088. "operation": o.operation,
3089. "right": **this**.getMathExpressionTree(sides[0]),
3090. "left": **null**
3091. }
3092. }
3093. **if** (sides.length == 2) {
3094. **return** {
3095. "operation": o.operation,
3096. "left": **this**.getMathExpressionTree(sides[0]),
3097. "right": **this**.getMathExpressionTree(sides[1])
3098. }
3099. }
3100. } **else** {
3101. **while** (isInBrackets(subline)) {
3102. subline = **this**.getLineFromFirstBrackets(subline).res;
3103. }
3104. **if** (!isNaN(parseFloat(subline))) {
3105. **return** parseFloat(subline);
3106. } **else** {
3107. **return** **this**.parseSimpleMathExpression(subline);
3108. }
3109. }
3110. }
3112. AT\_KRL\_Parser.prototype.getMathExpressionFromTree = **function** (root) {
3113. **if** (root.left != **null** && root.right != **null**) {
3114. **return** **new** AT\_KRL\_MathExpression([**this**.getMathExpressionFromTree(root.left), **this**.getMathExpressionFromTree(root.right)], root.operation.sing);
3115. }
3116. **if** (root.right != **null**) {
3117. **return** **new** AT\_KRL\_MathExpression([**this**.getMathExpressionFromTree(root.right)], root.operation.sing);
3118. }
3119. **if** (AT\_KRL\_MathExpression.prototype.isNumber(root)) {
3120. **return** **new** AT\_KRL\_MathExpression(root);
3121. }
3122. }
3124. AT\_KRL\_Parser.prototype.parseMathExpression = **function** (line) {
3125. **var** tree;
3126. **if** (line.replaceAll(' ', '') == '') {
3127. tree = **this**.getMathExpressionTree('0');
3128. } **else** {
3129. tree = **this**.getMathExpressionTree(line.replaceAll(' ', ''));
3130. }
3131. **return** **this**.getMathExpressionFromTree(tree);
3132. }
3134. //------Парсинг фактов для математических выражений и парсинг строковых или нечетких выражений------
3135. AT\_KRL\_Parser.prototype.parseStFzFact = **function** (line) {
3136. **while** (line.indexOf(' ') == 0) {
3137. line = line.slice(1);
3138. }
3139. **if** (line.indexOf('"') == 0) {
3140. **throw** **new** SyntaxError('Невозможно получить факт из объявления:\n' + line + '\nЕсли в выражении присутствует строка, то в левой части выражения должен быть объект и атрибут, а строка в правой');
3141. }
3142. **if** (line.indexOf('=') == -1 && line.indexOf('<') == -1 && line.indexOf('>') == -1) {
3143. **throw** **new** SyntaxError('Невозможно получить факт из объявления:\n' + line + '\nВыражения для символьных и нечетких значений должны сравниваться знаками "=", "<" и ">"')
3144. }
3145. **var** sings = ['=', '<', '>', '>=', '<='];
3146. **var** eq = -1;
3147. **var** sing = '=';
3148. **for** (**var** i = 0; i < sings.length; i++) {
3149. **var** tmp = line.indexOf(sings[i]);
3150. **if** (tmp != -1) {
3151. **if** ((tmp <= eq || eq == -1) && (sings[i].length == 1 || (sings[i].length == 2) && (tmp + 1 == line.indexOf('=')))) {
3152. eq = tmp;
3153. sing = sings[i];
3154. }
3155. }
3156. }
3157. **var** l = line.substring(0, eq);
3158. **var** r = line.substring(eq + sing.length, line.lastIndexOf(' УВЕРЕННОСТЬ ['));
3160. **var** left = **this**.parseStFzExpression(l);
3161. **var** right = **this**.parseStFzExpression(r);
3163. **var** ExSu = line.substring(line.lastIndexOf('УВЕРЕННОСТЬ ['), line.length);
3165. **function** checkExSu(str) {
3166. **var** s = str;
3167. **while** (s.charAt(s.length - 1) == ' ') {
3168. s = s.slice(0, s.length - 1);
3169. }
3170. **var** check = (s.indexOf('УВЕРЕННОСТЬ [') == 0 && s.indexOf('] ТОЧНОСТЬ ') != 0);
3171. **if** (!check) {
3172. **return** (s.indexOf('УВЕРЕННОСТЬ [') != 0) ? ' Некорректный символ на позиции 0\nТребуемый формат: УВЕРЕННОСТЬ [число >=0, число >=0] ТОЧНОСТЬ число >=0' : (s.indexOf(']') == -1 ? ' Не найден конец объявления уверенности в ' + s + '\nТребуемый формат: УВЕРЕННОСТЬ [число >=0, число >=0] ТОЧНОСТЬ число >=0' : ' Невозможно получить значение точности\nТребуемый формат: УВЕРЕННОСТЬ [число >=0, число >=0] ТОЧНОСТЬ число >=0');
3173. }
3174. **var** sure = s.substring(s.indexOf('['), s.indexOf('] ТОЧНОСТЬ ') + 1);
3175. **if** (sure.indexOf(';') != -1) {
3176. sure = sure.replaceAll(',', '.').replaceAll(';', ',');
3177. }
3178. **var** sr;
3179. **try** {
3180. sr = JSON.parse(sure);
3181. **if** (sr.length != 2) {
3182. **return** ' Невозможно получить значение уверенности в объявлении:\n' + s + '\nТребуемый формат: УВЕРЕННОСТЬ [число >=0, число >=0] ТОЧНОСТЬ число >=0';
3183. }
3184. **if** (**typeof** (sr[0]) != 'number' || **typeof** (sr[1]) != 'number') {
3185. **return** ' Невозможно получить значение уверенности в объявлении:\n' + s + '\nТребуемый формат: УВЕРЕННОСТЬ [число >=0, число >=0] ТОЧНОСТЬ число >=0';
3186. }
3187. } **catch** (e) {
3188. **return** ' Невозможно получить значение уверенности в объявлении:\n' + s + '\nТребуемый формат: УВЕРЕННОСТЬ [число >=0, число >=0] ТОЧНОСТЬ число >=0';
3189. }
3190. **var** exIndex = s.indexOf('] ТОЧНОСТЬ ') + ('] ТОЧНОСТЬ ').length;
3191. **for** (**var** i = exIndex; i < s.length; i++) {
3192. **var** count = 0;
3193. **if** (isNaN(parseInt(s.charAt(i))) && (s.charAt(i) != ',' || s.charAt(i) != '.') || (s.charAt(i) == ',' || s.charAt(i) == '.') && count > 0) {
3194. **return** ' Невозможно получить значение точности в объявлении:\n' + s + '\nНекорректный символ на позиции ' + i + '\nТребуемый формат: УВЕРЕННОСТЬ [число >=0, число >=0] ТОЧНОСТЬ число >=0';
3195. }
3196. **if** (s.charAt(i) == ',' || s.charAt(i) == '.') {
3197. count++;
3198. }
3199. }
3200. **return** -1;
3201. }
3203. **if** (line.indexOf('УВЕРЕННОСТЬ') == -1 || checkExSu(ExSu) != -1) {
3204. **throw** **new** SyntaxError('Невозможно получить значение уверенности и точности в объявлении:\n' + ExSu + '\n' + checkExSu(ExSu));
3205. }
3206. **var** s = ExSu.substring(ExSu.indexOf('['), ExSu.indexOf(']') + 1);
3207. **if** (s.indexOf(';') != -1) {
3208. s = s.replaceAll(',', '.').replaceAll(';', ',');
3209. }
3210. **var** sure = JSON.parse(s);
3212. **var** exact = parseFloat(ExSu.substring(ExSu.indexOf('] ТОЧНОСТЬ ') + ('] ТОЧНОСТЬ ').length, ExSu.length).replaceAll(',', '.'));
3213. **var** f = **new** AT\_KRL\_Fact(left, right, sing, sure, exact);
3214. **return** f;
3215. }
3217. AT\_KRL\_Parser.prototype.parseMathFact = **function** (line) {
3218. **while** (line.indexOf(' ') == 0) {
3219. line = line.slice(1);
3220. }
3222. **function** getRel(line) {
3223. **var** s = ['<=', '>=', '<', '>', '='];
3224. **for** (**var** i = 0; i < s.length; i++) {
3225. **if** (line.indexOf(s[i]) != -1) {
3226. **return** s[i];
3227. }
3228. }
3229. }
3231. **var** rel = getRel(line);
3232. **var** eq = line.indexOf(rel);
3234. **var** l = line.substring(0, eq);
3235. **while** (l.charAt(0) == '"') {
3236. l = l.slice(1);
3237. }
3238. **while** (l.charAt(l.length - 1) == '"') {
3239. l = l.slice(0, l.length - 1);
3240. }
3241. **var** r = line.substring(eq + rel.length, line.lastIndexOf(' УВЕРЕННОСТЬ ['));
3242. **while** (r.charAt(0) == '"') {
3243. r = r.slice(1);
3244. }
3245. **while** (r.charAt(r.length - 1) == '"') {
3246. r = r.slice(0, r.length - 1);
3247. }
3249. **var** left = **this**.parseMathExpression(l);
3250. **var** right = **this**.parseMathExpression(r);
3252. **var** ExSu = line.substring(line.lastIndexOf('УВЕРЕННОСТЬ ['), line.length);
3254. **function** checkExSu(str) {
3255. **var** s = str;
3256. **while** (s.charAt(s.length - 1) == ' ') {
3257. s = s.slice(0, s.length - 1);
3258. }
3259. **var** check = (s.indexOf('УВЕРЕННОСТЬ [') == 0 && s.indexOf('] ТОЧНОСТЬ ') != 0);
3260. **if** (!check) {
3261. **return** (s.indexOf('УВЕРЕННОСТЬ [') != 0) ? ' Некорректный символ на позиции 0\nТребуемый формат: УВЕРЕННОСТЬ [число >=0, число >=0] ТОЧНОСТЬ число >=0' : (s.indexOf(']') == -1 ? ' Не найден конец объявления уверенности в ' + s + '\nТребуемый формат: УВЕРЕННОСТЬ [число >=0, число >=0] ТОЧНОСТЬ число >=0' : ' Невозможно получить значение точности\nТребуемый формат: УВЕРЕННОСТЬ [число >=0, число >=0] ТОЧНОСТЬ число >=0');
3262. }
3263. **var** sure = s.substring(s.indexOf('['), s.indexOf('] ТОЧНОСТЬ ') + 1);
3264. **if** (sure.indexOf(';') != -1) {
3265. sure = sure.replaceAll(',', '.').replaceAll(';', ',');
3266. }
3267. **var** sr;
3268. **try** {
3269. sr = JSON.parse(sure);
3270. **if** (sr.length != 2) {
3271. **return** ' Невозможно получить значение уверенности в объявлении:\n' + s + '\nТребуемый формат: УВЕРЕННОСТЬ [число >=0, число >=0] ТОЧНОСТЬ число >=0';
3272. }
3273. **if** (**typeof** (sr[0]) != 'number' || **typeof** (sr[1]) != 'number') {
3274. **return** ' Невозможно получить значение уверенности в объявлении:\n' + s + '\nТребуемый формат: УВЕРЕННОСТЬ [число >=0, число >=0] ТОЧНОСТЬ число >=0';
3275. }
3276. } **catch** (e) {
3277. **return** ' Невозможно получить значение уверенности в объявлении:\n' + s + '\nТребуемый формат: УВЕРЕННОСТЬ [число >=0, число >=0] ТОЧНОСТЬ число >=0';
3278. }
3279. **var** exIndex = s.indexOf('] ТОЧНОСТЬ ') + ('] ТОЧНОСТЬ ').length;
3280. **for** (**var** i = exIndex; i < s.length; i++) {
3281. **var** count = 0;
3282. **if** (isNaN(parseInt(s.charAt(i))) && (s.charAt(i) != ',' || s.charAt(i) != '.') || (s.charAt(i) == ',' || s.charAt(i) == '.') && count > 0) {
3283. **return** ' Невозможно получить значение точности в объявлении:\n' + s + '\nНекорректный символ на позиции ' + i + '\nТребуемый формат: УВЕРЕННОСТЬ [число >=0, число >=0] ТОЧНОСТЬ число >=0';
3284. }
3285. **if** (s.charAt(i) == ',' || s.charAt(i) == '.') {
3286. count++;
3287. }
3288. }
3289. **return** -1;
3290. }
3292. **if** (line.indexOf('УВЕРЕННОСТЬ') == -1 || checkExSu(ExSu) != -1) {
3293. **throw** **new** SyntaxError('Невозможно получить значение уверенности и точности в объявлении:\n' + ExSu + '\n' + checkExSu(ExSu));
3294. }
3295. **var** s = ExSu.substring(ExSu.indexOf('['), ExSu.indexOf(']') + 1);
3296. **if** (s.indexOf(';') != -1) {
3297. s = s.replaceAll(',', '.').replaceAll(';', ',');
3298. }
3299. **var** sure = JSON.parse(s);
3300. **var** exact = parseFloat(ExSu.substring(ExSu.indexOf('] ТОЧНОСТЬ ') + ('] ТОЧНОСТЬ ').length, ExSu.length).replaceAll(',', '.'));
3301. **var** f = **new** AT\_KRL\_Fact(left, right, rel, sure, exact);
3302. **return** f;
3303. }
3305. //------Парсинг условий для правил методом построения и обхода дерева разбора------
3306. AT\_KRL\_Parser.prototype.getToNextLogicOperation = **function** (subline, sNames, index, current) {
3307. **var** i = index + current.length;
3308. **var** bracket = (subline.charAt(i) == '(') ? 1 : 0;
3310. **function** isOperation(sl, n, i) {
3311. **function** isInBrackets(line) {
3312. **if** (line.charAt(0) != '(') {
3313. **return** **false**;
3314. } **else** {
3315. **var** b = 1;
3316. **var** i = 1;
3317. **var** index = 0;
3318. **while** (b != 0 && i < line.length) {
3319. **if** (line.charAt(i) == '(') {
3320. b++;
3321. }
3322. **if** (line.charAt(i) == ')') {
3323. b--;
3324. }
3325. **if** (b == 0 && index == 0) {
3326. index = i;
3327. }
3328. **if** (b == 0 && i != line.length - 1) {
3329. **return** **false**;
3330. }
3331. i++;
3332. }
3333. **if** (b == 0 && index == line.length - 1) {
3334. **return** **true**;
3335. }
3336. }
3337. }
3339. **var** s = sl;
3340. **while** (isInBrackets(s)) {
3341. s = AT\_KRL\_Parser.prototype.getLineFromFirstBrackets(s).res;
3342. }
3343. **var** res = **false**;
3344. **for** (**var** k = 0; k < n.length; k++) {
3345. res = (res || ((s.indexOf(n[k], i) == i) && (n.indexOf(s.substr(i, n[k].length + 1)) == -1))) //&& (s.lastIndexOf(')') > s.lastIndexOf('УВЕРЕННОСТЬ') || s.lastIndexOf(')') == -1);
3346. **if** (res) {
3347. **return** n[k];
3348. }
3349. }
3350. **return** **false**;
3351. }
3352. **var** o = isOperation(subline, sNames, i);
3353. **while** ((!isOperation(subline, sNames, i) || isOperation(subline, sNames, i) && bracket != 0) && i < subline.length - 1) {
3354. i++;
3355. o = isOperation(subline, sNames, i);
3356. **if** (subline.charAt(i) == "(") {
3357. bracket++;
3358. }
3359. **if** (subline.charAt(i) == ")") {
3360. bracket--;
3361. }
3362. }
3363. **if** (i == subline.length - 1) {
3364. **return** **false**;
3365. }
3366. **return** {
3367. index: i,
3368. operation: o
3369. };
3370. }
3372. AT\_KRL\_Parser.prototype.getToFirstLogicOperation = **function** (subline, sNames) {
3373. **return** **this**.getToNextLogicOperation(subline, sNames, 0, '');
3374. }
3376. AT\_KRL\_Parser.prototype.getAllLogicSings = **function** () {
3377. **return** [{
3378. "sing": "~",
3379. "pos": 1,
3380. "type": ["coef"],
3381. "priority": [1]
3382. }, {
3383. "sing": "|",
3384. "pos": Infinity,
3385. "type": ["bin"],
3386. "priority": [0]
3387. }, {
3388. "sing": "&",
3389. "pos": Infinity,
3390. "type": ["bin"],
3391. "priority": [1]
3392. }]
3393. }
3395. AT\_KRL\_Parser.prototype.getLogicSingByName = **function** (name) {
3396. **var** s = **this**.getAllLogicSings();
3397. **for** (**var** i = 0; i < s.length; i++) {
3398. **if** (s[i].sing == name) {
3399. **return** s[i];
3400. }
3401. }
3402. }
3404. AT\_KRL\_Parser.prototype.getLogicOperationSides = **function** (subline, operation, index) {
3405. **function** isInBrackets(line) {
3406. **if** (line.charAt(0) != '(') {
3407. **return** **false**;
3408. } **else** {
3409. **var** b = 1;
3410. **var** i = 1;
3411. **var** index = 0;
3412. **while** (b != 0 && i < line.length) {
3413. **if** (line.charAt(i) == '(') {
3414. b++;
3415. }
3416. **if** (line.charAt(i) == ')') {
3417. b--;
3418. }
3419. **if** (b == 0 && index == 0) {
3420. index = i;
3421. }
3422. **if** (b == 0 && i != line.length - 1) {
3423. **return** **false**;
3424. }
3425. i++;
3426. }
3427. **if** (b == 0 && index == line.length - 1) {
3428. **return** **true**;
3429. }
3430. }
3431. }
3433. **var** left = **this**.getLineFromFirstBrackets(subline);
3434. **var** right = **this**.getLineFromBrackets(subline, index + 1);
3435. **while** (isInBrackets(left)) {
3436. left = **this**.getLineFromFirstBrackets(left).res;
3437. }
3438. **while** (isInBrackets(right)) {
3439. right = **this**.getLineFromFirstBrackets(right).res;
3440. }
3441. **if** (operation == '~') {
3442. **return** [right];
3443. }
3444. /\*if (){
3445. throw new SyntaxError('Невозможно интерпретировать математическое выражение в объявлении:\n' + subline);
3446. }\*/
3447. **return** [left, right];
3448. }
3450. AT\_KRL\_Parser.prototype.getLogicExpressionTree = **function** (subline) {
3451. **function** isInBrackets(line) {
3452. **if** (line.charAt(0) != '(') {
3453. **return** **false**;
3454. } **else** {
3455. **var** b = 1;
3456. **var** i = 1;
3457. **var** index = 0;
3458. **while** (b != 0 && i < line.length) {
3459. **if** (line.charAt(i) == '(') {
3460. b++;
3461. }
3462. **if** (line.charAt(i) == ')') {
3463. b--;
3464. }
3465. **if** (b == 0 && index == 0) {
3466. index = i;
3467. }
3468. **if** (b == 0 && i != line.length - 1) {
3469. **return** **false**;
3470. }
3471. i++;
3472. }
3473. **if** (b == 0 && index == line.length - 1) {
3474. **return** **true**;
3475. }
3476. }
3477. }
3478. **while** (isInBrackets(subline)) {
3479. subline = **this**.getLineFromFirstBrackets(subline).res;
3480. }
3481. subline = subline.replaceAll('\n', '');
3483. **var** sings = **this**.getAllLogicSings();
3484. **var** sNames = [];
3485. **for** (**var** i = 0; i < sings.length; i++) {
3486. **if** (sings[i].sing) {
3487. sNames.push(sings[i].sing);
3488. }
3489. }
3491. **var** thisLayerOperations = [];
3492. **var** o = **this**.getToFirstLogicOperation(subline, sNames);
3493. **while** (o) {
3494. thisLayerOperations.push({
3495. "operation": **this**.getLogicSingByName(o.operation),
3496. "index": o.index
3497. });
3498. o = **this**.getToNextLogicOperation(subline, sNames, o.index, o.operation);
3499. }
3501. **if** (thisLayerOperations.length != 0) {
3502. **var** o = thisLayerOperations[0];
3503. **for** (**var** i = 1; i < thisLayerOperations.length; i++) {
3504. **if** (thisLayerOperations[i].operation.priority[0] <= o.operation.priority[0]) {
3505. o = thisLayerOperations[i];
3506. }
3507. }
3508. **var** sides = **this**.getOperationSides(subline, o.operation, o.index);
3509. **if** (sides.length == 1) {
3510. **return** {
3511. "operation": o.operation,
3512. "right": **this**.getLogicExpressionTree(sides[0]),
3513. "left": **null**
3514. }
3515. }
3516. **if** (sides.length == 2) {
3517. **return** {
3518. "operation": o.operation,
3519. "left": **this**.getLogicExpressionTree(sides[0]),
3520. "right": **this**.getLogicExpressionTree(sides[1])
3521. }
3522. }
3523. } **else** {
3524. **try** {
3525. **return** **this**.parseMathFact(subline);
3526. } **catch** (e) {
3527. **try** {
3528. **return** **this**.parseStFzFact(subline);
3529. } **catch** (g) {
3530. **throw** **new** SyntaxError('ЕСЛИ ВЫ ХОТЕЛИ ВВЕСТИ ПРАВИЛО ДЛЯ ЧИСЛОВОГО ЗНАЧЕНИЯ:\n' + e.message + '\n\n' + 'ЕСЛИ ВЫ ХОТЕЛИ ВВЕСТИ ПРАВИЛО ДЛЯ СИМВОЛЬНОГО ИЛИ НЕЧЕТКОГО ЗНАЧЕНИЯ:\n' + g.message);
3531. }
3532. }
3533. }
3534. }
3536. AT\_KRL\_Parser.prototype.getConditionFromTree = **function** (root) {
3537. **var** cs = [];
3538. **if** (root.left != **null**) {
3539. **if** (root.left.operation != **null**) {
3540. cs.push(**this**.getConditionFromTree(root.left));
3541. }
3542. **if** (root.left.leftside != **null** && root.left.rightside != **null**) {
3543. cs.push(root.left);
3544. }
3545. }
3546. **if** (root.right.operation != **null**) {
3547. cs.push(**this**.getConditionFromTree(root.right));
3548. }
3549. **if** (root.right.leftside != **null** && root.right.rightside != **null**) {
3550. cs.push(root.right);
3551. }
3552. **var** s = root.operation.sing;
3553. **switch** (s) {
3554. **case** '~':
3555. **if** (cs.length == 1) {
3556. **return** **new** AT\_KRL\_ConditionNot(cs[0]);
3557. } **else** {
3558. **throw** **new** SyntaxError('Логический оператор "~" должен иметь единственный член');
3559. }
3560. **break**;
3561. **case** '&':
3562. **if** (cs.length == 2) {
3563. **return** **new** AT\_KRL\_ConditionAnd(cs);
3564. } **else** {
3565. **throw** **new** SyntaxError('Логический оператор "&" должен иметь минимум два члена');
3566. }
3567. **break**;
3568. **case** '|':
3569. **if** (cs.length == 2) {
3570. **return** **new** AT\_KRL\_ConditionOr(cs);
3571. } **else** {
3572. **throw** **new** SyntaxError('Логический оператор "|" должен иметь минимум два члена');
3573. }
3574. **break**;
3575. **default**:
3576. **throw** **new** SyntaxError('Ошибка чтения логического оператора из "' + s + '"');
3577. **break**;
3578. }
3579. }
3581. //------Парсинг правила------
3582. AT\_KRL\_Parser.prototype.parseRuleName = **function** (line) {
3583. **if** (line.indexOf("ПРАВИЛО ") != 0) {
3584. **throw** **new** SyntaxError("Невозможно получить имя правила из: " + line);
3585. }
3586. **var** name = **this**.validateName(line.replace('ПРАВИЛО ', ''), 'rules');
3587. **return** name;
3588. }
3590. AT\_KRL\_Parser.prototype.parseRuleNameFL = **function** (lines) {
3591. **return** **this**.parseRuleName(lines[0]);
3592. }
3594. AT\_KRL\_Parser.prototype.removeBracketSpases = **function** (line) {
3595. **return** line.replaceAll(' (', '(').replaceAll('( ', '(').replaceAll(' )', ')').replaceAll(') ', ')').replaceAll(')УВЕРЕННОСТЬ', ') УВЕРЕННОСТЬ');
3596. }
3598. AT\_KRL\_Parser.prototype.parseRuleConditions = **function** (line) {
3599. **var** tree = **this**.getLogicExpressionTree(**this**.removeBracketSpases(line));
3600. **return** **this**.getConditionFromTree(tree);
3601. }
3603. AT\_KRL\_Parser.prototype.parseRuleConditionsFL = **function** (lines) {
3604. **var** line = '';
3605. **for** (**var** i = lines.indexOf('ЕСЛИ') + 1; i < lines.indexOf('ТО'); i++) {
3606. line += lines[i];
3607. }
3608. **return** **this**.parseRuleConditions(line);
3609. }
3611. AT\_KRL\_Parser.prototype.sideHasNoObject = **function** (side) {
3612. **if** (**typeof** (side) == "number" || **typeof** (side) == "string") {
3613. **return** **true**;
3614. }
3615. **if** (side.expressions) {
3616. **var** res = **true**;
3617. **for** (**var** i = 0; i < side.expressions.length; i++) {
3618. res = res && **this**.sideHasNoObject(side.expressions[i]);
3619. }
3620. **return** res;
3621. }
3622. **return** **false**;
3623. }
3625. AT\_KRL\_Parser.prototype.sideIsOnlyObject = **function** (side) {
3626. **if** (side.object && side.aIndex != **null** && side.object.attributes[side.aIndex]) {
3627. **return** **true**;
3628. }
3629. **if** (side.expressions && side.expressions.length == 1) {
3630. **return** **this**.sideIsOnlyObject(side.expressions[0]);
3631. }
3632. **return** **false**;
3633. }
3635. AT\_KRL\_Parser.prototype.factIsSimple = **function** (f, l) {
3636. **var** res = **this**.sideIsOnlyObject(f.leftside) && **this**.sideHasNoObject(f.rightside);
3637. **if** (!res) {
3638. **throw** **new** SyntaxError('Описание факта "' + l + '" не подходит для описания результата правила\nТребуемый формат: ОБЪЕКТ.АТРИБУТ=значение');
3639. }
3640. **return** res;
3641. }
3643. AT\_KRL\_Parser.prototype.parseRuleFactsFL = **function** (lines) {
3644. **var** i1 = lines.indexOf('ТО');
3645. **var** i2 = lines.indexOf('ИНАЧЕ');
3646. **var** i3 = -1;
3647. **for** (**var** i = 0; i < lines.length; i++) {
3648. **if** (lines[i].indexOf('КОММЕНТАРИЙ') == 0) {
3649. i3 = i;
3650. }
3651. }
3652. **if** (i2 == -1) {
3653. i2 = i3;
3654. }
3655. **if** (i1 == -1 || i2 == -1) {
3656. **throw** **new** SyntaxError('Невозможно получить результат правила в объявлении:\n' + **this**.getTextFromLines(lines));
3657. }
3658. **var** result = [];
3659. **for** (**var** i = i1 + 1; i < i2; i++) {
3660. **var** f;
3661. **try** {
3662. f = **this**.parseMathFact(lines[i]);
3663. } **catch** (e) {
3664. **try** {
3665. f = **this**.parseStFzFact(lines[i]);
3666. } **catch** (g) {
3667. **throw** **new** SyntaxError('ЕСЛИ ВЫ ХОТЕЛИ ВВЕСТИ ПРАВИЛО ДЛЯ ЧИСЛОВОГО ЗНАЧЕНИЯ:\n' + e.message + '\n\n' + 'ЕСЛИ ВЫ ХОТЕЛИ ВВЕСТИ ПРАВИЛО ДЛЯ СИМВОЛЬНОГО ИЛИ НЕЧЕТКОГО ЗНАЧЕНИЯ:\n' + g.message);
3668. }
3669. }
3670. **if** (**this**.factIsSimple(f, lines[i])) {
3671. result.push(f);
3672. }
3673. }
3675. **var** elsresult = **null**;
3677. **if** (i2 != i3) {
3678. elsresult = [];
3679. **for** (**var** i = i2 + 1; i < i3; i++) {
3680. **var** f;
3681. **try** {
3682. f = **this**.parseMathFact(lines[i]);
3683. } **catch** (e) {
3684. **try** {
3685. f = **this**.parseStFzFact(lines[i]);
3686. } **catch** (g) {
3687. **throw** **new** SyntaxError('ЕСЛИ ВЫ ХОТЕЛИ ВВЕСТИ ПРАВИЛО ДЛЯ ЧИСЛОВОГО ЗНАЧЕНИЯ:\n' + e.message + '\n\n' + 'ЕСЛИ ВЫ ХОТЕЛИ ВВЕСТИ ПРАВИЛО ДЛЯ СИМВОЛЬНОГО ИЛИ НЕЧЕТКОГО ЗНАЧЕНИЯ:\n' + g.message);
3688. }
3689. }
3690. **if** (**this**.factIsSimple(f, lines[i])) {
3691. elsresult.push(f);
3692. }
3693. }
3694. }
3695. **return** {
3696. "result": result,
3697. "elsresult": elsresult
3698. };
3699. }
3701. AT\_KRL\_Parser.prototype.parseRule = **function** (declaration) {
3702. **var** lines = **this**.getAllLines(declaration);
3703. **var** name = **this**.parseRuleNameFL(lines);
3704. **var** conditions = **this**.parseRuleConditionsFL(lines);
3705. **var** facts = **this**.parseRuleFactsFL(lines);
3706. **var** comment = **this**.parseCommentFL(lines);
3707. **var** rule = **new** AT\_KRL\_Rule(name, conditions, facts, comment, **this**.editor);
3708. **return** rule;
3709. }
3711. //------Парсинг объявления------
3712. AT\_KRL\_Parser.prototype.parseDeclaration = **function** (declaration) {
3713. **var** lines = **this**.getAllLines(declaration);
3714. **var** decType = lines[0].slice(0, lines[0].indexOf(' ') + 1);
3715. **switch** (decType) {
3716. **case** 'ТИП ':
3717. **return** **this**.parseType(declaration);
3718. **case** 'ОБЪЕКТ ':
3719. **return** **this**.parseObject(declaration);
3720. **case** 'ПРАВИЛО ':
3721. **return** **this**.parseRule(declaration);
3722. **default**:
3723. **throw** **new** SyntaxError('Некорректное объявление:\n' + declaration + '\nСтрока 0');
3724. }
3725. }
3726. TKBConvertor.pas
3727. **unit** KBConvertor;
3729. //////////////////////////////////////////////////////////////
3730. ///  Модуль для конвертации файлов БЗ из .kbs в файлы .xml ///
3731. //////////////////////////////////////////////////////////////
3732. /// Использую                                              ///
3733. ///   https://github.com/grigandal625/AT\_KRL\_Editor        ///
3734. ///   JS-библиотека,                                       ///
3735. ///   интегрирую через ActiveX-компонент MSScriptControl   ///
3736. //////////////////////////////////////////////////////////////
3737. ///\*\*\*\*\*\*\*\*\*\*\*\*Автор: Григорьев А.А. 05.06.2019\*\*\*\*\*\*\*\*\*\*\*\*///
3738. //////////////////////////////////////////////////////////////
3740. **interface**
3742. **uses**
3743. Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
3744. System.Classes, Vcl.Graphics,
3745. Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ComCtrls, Vcl.StdCtrls, ComObj,
3746. Vcl.OleCtrls;
3748. **type**
3750. decs = **array** **of** **string**;
3752. TForm2 = **class**(TForm) // просто формочка со строкой прогресса конвертации
3753. Label1: TLabel;
3754. ProgressBar1: TProgressBar;
3755. Memo1: TMemo;
3756. StatusBar1: TStatusBar;
3757. **procedure** FormCreate(Sender: TObject);
3758. **private**
3759. { Private declarations }
3760. **public**
3761. { Public declarations }
3762. **end**;
3764. TKBConvertor = **class**(TObject)
3765. **private**
3766. js: OleVariant;
3767. declarations: decs;
3768. KBS: **string**;
3769. XML: **string**;
3770. **function** replaceAll(s, old, new: **string**): **string**;
3771. **function** getDeclarationsCount(s: **string**): **integer**;
3772. **function** getDeclarations(s: **widestring**): decs;
3773. **public**
3774. savePath: **string**;
3775. ini: TForm2;
3776. **constructor** create;
3777. **procedure** LoadFromFile(path: **string**; e: TEncoding);
3778. **procedure** SaveToFile(path: **string**);
3779. **procedure** Convert(path: **string**);
3780. **destructor** Destroy;
3782. **end**;
3784. **implementation**
3786. {$R \*.dfm}
3788. **procedure** TForm2.FormCreate(Sender: TObject);
3789. **begin**
3790. Caption := 'Статус';
3791. BorderIcons := [];
3792. BorderStyle := bsNone;
3793. Label1.layout := tlCenter;
3794. Label1.alignment := taLeftJustify;
3795. Label1.Caption := 'Инициализация';
3796. Position := poDesktopCenter;
3797. **end**;
3799. **constructor** TKBConvertor.create;
3800. **var**
3801. loader: TStringList;
3802. script: **string**;
3803. **begin**
3805. { Scripts: https://github.com/grigandal625/AT\_KRL\_Editor/tree/master/Scripts }
3806. ini := TForm2.create(application);
3807. loader := TStringList.create;
3808. js := CreateOleObject('ScriptControl'); // msscript.ocx
3809. js.Language := 'JScript';
3810. js.TimeOut := -1;
3811. // всегда ожидаем до конца, не останавливая выполнение скрипта
3812. js.AllowUI := True;
3814. // Загрузим все скрипты
3815. **try**
3817. loader.LoadFromFile('Scripts/JSUtils.js');
3818. script := loader.text;
3819. js.eval(script);
3820. loader.clear;
3821. loader.LoadFromFile('Scripts/XMLUtils.js');
3822. script := loader.text;
3823. js.eval(script);
3824. loader.clear;
3825. loader.LoadFromFile('Scripts/AT\_KRL\_Editor.js');
3826. script := loader.text;
3827. js.eval(script);
3828. loader.clear;
3829. loader.LoadFromFile('Scripts/AT\_KRL\_Type.js');
3830. script := loader.text;
3831. js.eval(script);
3832. loader.clear;
3833. loader.LoadFromFile('Scripts/AT\_KRL\_Object.js');
3834. script := loader.text;
3835. js.eval(script);
3836. loader.clear;
3837. loader.LoadFromFile('Scripts/AT\_KRL\_Expressions.js');
3838. script := loader.text;
3839. js.eval(script);
3840. loader.clear;
3841. loader.LoadFromFile('Scripts/AT\_KRL\_Fact.js');
3842. script := loader.text;
3843. js.eval(script);
3844. loader.clear;
3845. loader.LoadFromFile('Scripts/AT\_KRL\_Rule.js');
3846. script := loader.text;
3847. js.eval(script);
3848. loader.clear;
3849. loader.LoadFromFile('Scripts/AT\_KRL\_Parser.js');
3850. script := loader.text;
3851. js.eval(script);
3852. loader.clear;
3853. js.eval('var e = new AT\_KRL\_Editor(); var p = new AT\_KRL\_Parser(e); e.clear();');// e - объект, где будет сохраняться все распарсенное
3855. **except**
3856. **on** Er : Exception **do**
3857. **begin**
3858. inputBox('Ошибка конвертации', Er.ClassName+#13#10+#13#10+'Ошибка с сообщением : '+#13#10+replaceAll(Er.Message,'. ','.'+#13#10) + #13#10+#13#10+'(Если ошибка об отсутсвии файлов, загрузите их по ссылке.)', 'https://github.com/grigandal625/AT\_KRL\_Editor/tree/master/Scripts');
3859. ini.Close;
3860. exit;
3861. **end**;
3863. **end**;
3864. **end**;
3866. **destructor** TKBConvertor.Destroy;
3867. **begin**
3868. js.free;
3869. **end**;
3871. **function** TKBConvertor.replaceAll(s, old, new: **string**): **string**;
3872. **var**
3873. before, after: **string**;
3874. **begin**
3875. before := s;
3876. after := StringReplace(before, old, new, [rfReplaceAll]);
3877. result := after;
3878. **end**;
3880. // Считат, сколько всего надо распарсить в .kbs
3881. **function** TKBConvertor.getDeclarationsCount(s: **string**): **integer**;
3882. **var**
3883. KRL: **string**;
3884. **begin**
3885. KRL := s;
3886. result := StrToInt(js.eval('(p.getAllDeclarations(''' + replaceAll(KRL, '''',
3887. '\''') { Экранируем кавычки } + ''').length)'));
3888. **end**;
3890. // Получаем по-отдельности то, что надо распарсить
3891. **function** TKBConvertor.getDeclarations(s: **widestring**): decs;
3892. **var**
3893. res: decs;
3894. i, len: **integer**;
3895. KRL: **widestring**;
3896. **begin**
3897. len := getDeclarationsCount(s);
3898. KRL := '''' + replaceAll(s, '''', '\''') + '''';
3899. js.eval('var tmp = p.getAllDeclarations(' + KRL + ');');
3900. setLength(res, len);
3901. **for** i := 0 **to** len - 1 **do**
3902. **begin**
3903. res[i] := replaceAll(js.eval('JSON.stringify(tmp[' + IntToStr(i) + '])'),
3904. #10, #13#10); // replaceAll тут по сути и не нужен
3905. **end**;
3906. result := res;
3907. **end**;
3909. **procedure** TKBConvertor.LoadFromFile(path: **string**; e: TEncoding);
3910. **var**
3911. KRL: **string**;
3912. **begin**
3913. ini.Memo1.lines.LoadFromFile(path, e);
3914. KRL := replaceAll(ini.Memo1.text, #13#10, '\n');
3915. KBS := ini.Memo1.text;
3916. declarations := getDeclarations(KRL);
3917. **end**;
3919. **procedure** TKBConvertor.SaveToFile(path: **string**);
3920. **var**
3921. f: textFile;
3922. **begin**
3923. AssignFile(f, path);
3924. rewrite(f);
3925. **writeln**(f, XML);
3926. closefile(f);
3927. **end**;

3930. // конвертим из файла
3931. **procedure** TKBConvertor.Convert(path: **string**);
3932. **var**
3933. i, len: **integer**;
3934. s, er, scr, KRL: **string**;
3935. **begin**
3936. ini.Show;
3937. ini.ProgressBar1.Position := 0;
3938. application.ProcessMessages;
3939. **try**
3940. LoadFromFile(path, TEncoding.UTF8); // пробуем загрузить c UTF8
3941. **except**
3942. LoadFromFile(path, Nil); // Если не выходит, то с ANSI
3943. **end**;
3945. // Во время LoadFromFile уже получили список declarations
3947. len := length(declarations);
3948. ini.ProgressBar1.Max := len + len **div** 50;
3950. // парсим каждую из declarations, распарсенное сохранится в "e" (см Конструктор, последний js.eval)
3951. **for** i := 0 **to** len - 1 **do**
3952. **begin**
3953. s := declarations[i];
3954. **if** s <> '' **then**
3955. **begin**
3956. scr := 'try{p.parseDeclaration(' + AnsiToUTF8(s) +
3957. ');}catch(error){"!!!ERROR!!!: " + error.message}';
3958. er := js.eval(scr);
3959. **if** pos('!!!ERROR!!!', er) <> 0 **then**
3960. **begin**
3961. showMessage(UTF8ToAnsi(er));
3962. exit
3963. **end**;
3964. ini.ProgressBar1.Position := i;
3965. application.ProcessMessages;
3967. **end**;
3968. **end**;
3969. ini.Label1.Caption := 'Конвертация';
3970. application.ProcessMessages;
3972. //Конвертируем в XML из "e"
3973. XML := js.eval('e.toXML();');
3974. SaveToFile(savePath);
3975. ini.close;
3976. **end**;
3978. **end**.

Использование:

1. **procedure** TSolverX.Configurate(**const** Config: WideString);
2. **var**
3. cvt: TKBConvertor;
4. **begin**
5. {...
6. исходный код
7. ...}
8. **if** AnsiCompareStr(ExtractFileExt(Debugger.Solver.KBFileName), '.kbs') = 0 **then**
9. **begin**
10. cvt := TKBConvertor.create;
11. fn := ExtractFilePath(fn) + '~temp.xml';
12. cvt.savePath := fn;
13. cvt.Convert(Debugger.Solver.KBFileName);
14. **end**;
16. {...
17. исходный код
18. ...}
19. **end**;

Исправление инициализации нечеткого типа:

1. **procedure** TXMLLoader.LoadTypes(N: IXMLNode);
2. **var**
3. i, j, k: **integer**;
4. t: TKBType;
5. Meta, attrX, attrY: WideString;
6. M, PS, PM: IXMLNode;
7. Val: Variant;
8. MF: TMembershipFunction;
9. L: TList;
10. TMP: TFuzzyType;
11. **begin**
12. {...
13. Исходный код
14. ...}
16. **else** **if** AnsiCompareText(Meta, sFuzzy) = 0 **then**
17. **begin**
18. TMP := TFuzzyType.Create(**nil**);
19. L := TList.Create; //Сюда будем класть функции принадлежности ЛП
20. **with** TMP **as** TFuzzyType **do**
21. **begin**
22. **for** j := 0 **to** M.ChildNodes.Count - 1 **do** //пробегаемся по элементам <parametr>, из каждого создаем ФП и формируем список из ФП
23. **begin**
24. MF := TMembershipFunction.Create(Nil); //Создаем ФП, ее координаты - элементы <mf><point X="../>..<point X="../></mf>
25. PM := M.ChildNodes[j];
27. MF.MFName := (PM.ChildNodes['value'].Text);
28. PS := PM.ChildNodes['mf'];
29. **for** k := 0 **to** PS.ChildNodes.Count - 1 **do** //Добавляем координаты точек в кусочную ФП из <point>
30. **begin**
31. attrX := stringReplace((PS.ChildNodes[k].Attributes['X']),
32. '.', ',', []);
33. attrY := stringReplace((PS.ChildNodes[k].Attributes['Y']),
34. '.', ',', []);
35. MF.AddXY(strtofloat(attrX), strtofloat(attrY));
36. **end**;
37. L.Add(MF);
38. **end**;
39. TMP.MFList := L;
40. t := TMP;
41. **end**;
42. **end**
44. {...
45. Исходный код
46. ...}
47. **end**;